CA**BIO** 4.3

Technical Guide

Center for Biomedical Informatics
and Information Technology

July 16, 2009

CREDITS AND RESOURCES

caBIO Development and Management Teams		
Development	**Technical Guide**	**Program Management**
Hong Dang [3]	Carolyn Kelley Klinger[5]	Juli Klemm[1]
Sharon Gaheen[2]	Konrad Rokicki[2]	Krishnakant Shanbhag[1]
Norval Johnson[4]	Jim Sun[2]	George Komatsoulis[1]
Sriram Kalyanasundaram[4]	Lalitha Viswanath[2]	
Ying Long[2]	Sharon Gaheen[2]	
Konrad Rokicki[2]		
Jim Sun[2]		
Lalitha Viswanath[2]		
Ye Wu[2]		
Tom Boal[6]		
[1] National Cancer Institute Center for Biomedical Informatics and Information Technology (NCI CBIIT)		[2] Science Applications International Corporation (SAIC)
[3] Alpha Gamma Technologies		[4] TerpSys
[5] Lockheed Martin Management System Designers		[6] NARTech

Contacts and Support	
Application Support	http://ncicb.nci.nih.gov/NCICB/support
	Telephone: 301-451-4384
	Toll free: 888-478-4423

LISTSERV Facilities Pertinent to caBIO		
LISTSERV	*URL*	*Name*
CABIO_USERS	https://list.nih.gov/archives/cabio_users.html	Users
CABIO_DEVELOPERS	https://list.nih.gov/archives/cabio_developers.html	Developers

GForge is a cross project collaboration site for NCI CBIIT caBIO developers located at http://gforge.nci.nih.gov/projects/cabiodb/.

Release Schedule

This guide is updated for each caBIO release. It may be updated between releases if errors or omissions are found. The current document refers to the 4.3 version of caBIO, which was released in July 2009 by NCI CBIIT.

CONTENTS

ABOUT THIS GUIDE

This section introduces you to the *caBIO 4.3 Technical Guide*. Topics in this section include:

- *Purpose* on this page

- *New Features in caBIO 4.3* on page 2

- *Audience* on page 3

- *Additional caBIO Documentation* on page 3

- *Organization of This guide* on page 3

- *Text Conventions Used* on page 4

Purpose

The *caBIO 4.3 Technical Guide* describes the Cancer Bioinformatics Infrastructure Objects (caBIO) model, the data represented by its objects, and its application programming interfaces.

This guide describes:

- Purpose, architecture and components of caBIO

- APIs for accessing the caBIO system including Java, Web services, and XML-HTTP

- Additional utilities for accessing the data in caBIO

- Data accessible through the caBIO APIs

Tip: Code snippets in this and other chapters of the technical guide PDF can be copied for pasting elsewhere. Right-click anywhere in the PDF and select **Allow Hand Tool to Select Text**. Click and drag to select text. Copy the text, then paste the text into NotePad or WordPad to retain the line distinctions.

New Features in caBIO 4.3

The following are the new features of caBIO 4.3:

- Enhancements to the caBIO agent and associated objects in support of data from the Canada DrugBank.

- Inclusion of objects modeling highly conserved regions on the human genome and support for multiple-alignment data from Ensembl Compara.

- Inclusion of annotation version field to the Microarray class in order to expose the version of Affymetrix annotations.

- Upgrade to the caCORE SDK 4.1.1.

The following enhancements to the caBIO Graphical User Interface (GUI) were also implemented:

- caBIO Home Page:

    - Added auto-complete feature to the FreeStyleLM ("Google-like" search) search option.

    - Added a new caBIO build tag.

- caBIO Portlet:

    - Added auto-complete feature to the Simple ("Google-like" search) search option.

    - Improved advanced search options by including the ability to configure the number of results per page.

    - Added new templated searches including Pathway Search by Source and Protein and Gene Ontology Search by Protein.

    - Made general usability enhancements including adding an About tab that displays detailed version information and modifying the layout to run in 50% of the screen for optimal coexistence with future portlets.

Audience

The primary audiences of this guide are:

- The application developer who wants to learn about the architecture and use of the caBIO APIs and utilities. These sections of the guide assume familiarity with the Java programming language and/or other programming languages, database concepts, and the Internet. If you intend to use caCORE resources in software applications, it assumes that you have experience with building and using complex data systems.

- The scientist end-user who wants to understand the data provided in caBIO (described in *Chapter 2*) and/or use the Freestyle Lexical Mine search (described in *Chapter 6*).

Additional caBIO Documentation

- The caBIO 4.3 Release Notes contain a description of the end user tool enhancements and bug fixes included in this release.

- The caBIO 4.3 JavaDocs contain the current caBIO API specification

- The caCORE SDK 4.1 Developer's Guide contains detailed instructions on the use of the SDK and how it helps create a caCORE-like software system.

- For more caBIO-related references, see *Appendix A*.

Organization of This guide

This brief overview explains what you will find in each section of this guide.

- This chapter introduces this technical guide.

- *Chapter 1, caCORE Overview,* on page 5 provides an overview of caCORE and caBIG®.

- *Chapter 2, caBIO Overview,* on page 9 provides an overview of caBIO.

- *Chapter 3, caBIO Architecture,* on page 13 describes the architecture of caBIO.

- *Chapter 4, Interacting with caBIO,* on page 19 describes the various methods for accessing the data in caBIO.

- *Chapter 6, caBIO Utilities,* on page 83 describes additional utilities provided by the caBIO API, which are not generated by the caCORE SDK.

- *Chapter 5, Cancer Bioinformatics Infrastructure Objects,* on page 57 describes the Cancer Bioinformatics Infrastructure Objects (caBIO).

- *Appendix A, References*, on page 97 includes a listing of references to provide the caBIO users with more in depth information about relevant topics.

Text Conventions Used

This section explains conventions used in this guide. The various typefaces represent interface components, keyboard shortcuts, toolbar buttons, dialog box options, and text that you type.

Convention	*Description*	*Example*
Bold	Highlights names of option buttons, check boxes, drop-down menus, menu commands, command buttons, or icons.	Click **Search**.
URL	Indicates a Web address.	http://domain.com
text in SMALL CAPS	Indicates a keyboard shortcut.	Press ENTER.
text in SMALL CAPS + text in SMALL CAPS	Indicates keys that are pressed simultaneously.	Press SHIFT + CTRL.
Italics	Highlights references to other documents, sections, figures, and tables.	See *Figure 4.5.*
Italic boldface monospaced type	Represents text that you type.	In the **New Subset** text box, enter *Proprietary Proteins.*
Note:	Highlights information of particular importance	**Note:** This concept is used throughout the document.
{ }	Surrounds replaceable items.	Replace {last name, first name} with the Principal Investigator's name.

1

CACORE OVERVIEW

This chapter provides an overview of the CBIIT caCORE infrastructure.

Topics in this chapter include:

- *Architecture Overview* on this page
- *Components of caCORE* on page 6

Architecture Overview

The NCI Center for Biomedical Informatics and Information Technology (CBIIT) provides biomedical informatics support and integration capabilities to the cancer research community. NCI CBIIT has created a core infrastructure called Cancer Common Ontologic Representation Environment (caCORE), a data management framework designed for researchers who need to be able to navigate through a large number of data sources. By providing a common data management framework, caCORE helps streamline the informatics development throughout academic, government and private research labs and clinics. The components of caCORE support the semantic consistency, clarity, and comparability of biomedical research data and information. caCORE is open-source enterprise architecture for NCI-supported research information systems, built using formal techniques from the software engineering and computer science communities. The four characteristics of caCORE include:

- Model Driven Architecture (MDA)
- *n*-tier architecture with open Application Programming Interfaces (APIs)
- Use of controlled vocabularies, wherever possible
- Registered metadata

The use of MDA and n-tier architecture, both standard software engineering practices, allows for easy access to data, particularly by other applications. The use of controlled

vocabularies and registered metadata, less common in conventional software practices, requires specialized tools, generally unavailable.

As a result, CBIIT (in cooperation with the NCI Office of Communications) has developed the Enterprise Vocabulary Services (EVS) system to supply controlled vocabularies, and the Cancer Data Standards Repository (caDSR) to provide a dynamic metadata registry. When all four development principles are addressed, the resulting system has several desirable properties. Systems with these properties are said to be "caCORE-like".

1. The *n*-tier architecture with its open APIs frees the end user (whether human or machine) from needing to understand the implementation details of the underlying data system to retrieve information.

2. The maintainer of the resource can move the data or change implementation details (Relational Database Management System, and so forth) without affecting the ability of remote systems to access the data.

Most importantly, the system is 'semantically interoperable'; that is, there exists runtime-retrievable information that can provide an explicit definition and complete data characteristics for each object and attribute that can be supplied by the data system.

Components of caCORE

The components that comprise caCORE are EVS, caDSR, caBIO, CSM, and CLM. Each is described briefly below.

Enterprise Vocabulary Services (EVS)

EVS provides controlled vocabulary resources that support the life sciences domain, implemented in a description logics framework. EVS vocabularies provide the semantic 'raw material' from which data elements, classes, and objects are constructed.

Cancer Data Standards Repository (caDSR)

The caDSR is a metadata registry, based upon the ISO/IEC 11179 standard, used to register the descriptive information needed to render cancer research data reusable and interoperable. The caBIO, EVS, and caDSR data classes are registered in the caDSR, as are the data elements on NCI-sponsored clinical trials case report forms.

caCORE Software Development Toolkit (SDK)

The caCORE SDK is a model-driven software engineering tool for creating caCORE-like software systems. The software generated by the SDK is combined with controlled vocabularies and registered metadata to facilitate the creation of semantically interoperable systems. caBIO leverages the caCORE SDK to generate the Java API, web services, and the REST API.

Common Security Model (CSM)

CSM provides a flexible solution for application security and access control with three main functions:

- Authentication to validate and verify a user's credentials

- Authorization to grant or deny access to data, methods, and objects

- User Authorization Provisioning to allow an administrator to create and assign authorization roles and privileges.

Common Logging Module (CLM)

CLM provides a separate service under caCORE for Audit and Logging Capabilities. It also comes with a web based locator tool. It can be used by a client application directly, without the application using any other components like CSM.

CABIO OVERVIEW

This chapter provides an overview of the caBIO data and infrastructure.

Topics in this chapter include:

- *General Overview* on this page
- *caBIO Data Overview* on this page
- *Architecture Overview* on page 11

General Overview

The cancer Bioinformatics Infrastructure Objects (caBIO) model and architecture ws the first model-driven information system built using caBORE, a synthesis of software, vocabulary, and metadata models for cancer research. Each of the caBIO domain objects represents an entity found in biomedical research such as Gene, Chromosome, Sequence, SNP, Library, Clone, and Pathway. Examining the relationships between these objects can reveal biomedical knowledge that was previously buried in the various primary data sources. Given the dynamic nature of this information, the data in caBIO is updated on a monthly basis through a series of ETL (Extract, Transform, and Load) processes.

The caBIO data can be accessed via a variety of Application Programming Interfaces (APIs) as well as a web user interface called the caBIO Portlet, which you can access through the caGrid Portal (for more information on the caGrid Portal, see http://cagrid-portal.nci.nih.gov/). The system also provides a number of utilities to search for and manipulate data objects programmatically.

caBIO Data Overview

The caBIO system is primarily a repository of molecular biology data; the entities that concern the central dogma of molecular biology are the core of this model. Data for these classes come from a variety of sources. Gene, Chromosome, Nucleic Acid Sequence, and Taxon comes from the *NCBI* UniGene data resource directly as well as via the Cancer Gene Anatomy Project (CGAP), an NCI activity that maintains information about expression of genes in tumors based on EST sequencing frequency. Cytogenetic location information, as well as EST and mRNA sequence annotation data, come from the University of California Santa Cruz (UCSC) Genome Project. Protein data (including data about sequences) is retrieved from the Protein Information Resource (PIR) at Georgetown University. For a full list of primary data sources, see https://wiki.nci.nih.gov/display/ICR/caBIO+Data+Sources.

Pathway data in caBIO comes from the Pathway Interaction Database (PID), a highly-structured, curated collection of information about known biomolecular interactions and key cellular processes assembled into signaling pathways. The PID is a collaborative effort between the National Cancer Institute (NCI) and Nature Publishing Group (NPG), and is an open access online resource. The caBIO domain model provides classes representing PID components and interactions.

Annotations from the most widely used human microarrays from Affymetrix (http://www.affymetrix.com), Illumina (http://www.illumina.com), and Aglient (http://www.agilent.com) are stored in caBIO, to allow the rich annotations provided by the manufacturers to be integrated with the other caBIO data sources and served through a caCORE interface. Data for these microarray annotations is distributed across many of the caBIO classes; the core classes are `Microarray` and `ArrayReporter` with subclasses `SNPArrayReporter`, `ExpressionArrayReporter`, and `TranscriptArrayReporter`.

Single Nucleotide Polymorphism data are primarily supplied by dbSNP and by Affymetrix (http://www.affymetrix.com). The core classes are `SNP`, `PopulationFrequency`, and `GeneRelativeLocation`.

The caBIO system provides information about libraries used as part of EST sequencing efforts. This set of classes (which includes Library, Tissue, and Protocol) are populated with data from the NCI CGAP initiative. This group of classes provides an instructive example about the necessity of grounding metadata in concept-based controlled terminology. The Protocol object here represents the method used to create a library from a tissue (NCI Thesaurus concept code C41111) rather than the concept of a clinical trial protocol (C25320).

Clinical trial data in the caBIO system are obtained from the Cancer Therapeutics Evaluation Program (CTEP) of the NCI. This part of the model was originally driven by use cases from the NCI CMAP project, and as such, the classes contain top level about ongoing clinical trials. Constrained regions on the genome are added from Ensembl's Compara database. Additional drug compound data comes from the Canada Drugbank and includes pharmacologic and pharmacokinetic attributes as well as genetic drug targets.

Architecture Overview

The caBIO system is an n-tier client-server architecture. The caBIO application server, located at NCI (http://cabioapi.nci.nih.gov/cabio43), provides a home page for web-based searching and data retrieval, as well as access to a wide variety of APIs for programmatic access.

The caBIO server interacts with a database backend on behalf of the user to serve requested objects to the client. Hibernate-based object-relational mapping (ORM) is used to populated caBIO domain objects. Only object-oriented APIs are exposed to clients so that the relational database backend is completely transparent to the user.

Many of the APIs rely on a query-by-example mechanism that takes a partially-populated object in, and returns other objects like it back to the user. This paradigm of interaction makes semantic interoperability much easier.

The following APIs are available for programmatic data access:

- Java API

- Web Services API

- REST API

- Grid Service (The current grid service is for caBIO 3.2 and 4.0.)

The system also provides several additional utilities for accessing the data in caBIO:

- Freestyle Lexical Mine (FreestyleLM) search

- Genome Range Queries

- Grid ID Resolver

- Array Annotation APIs

- XML Marshalling Utility for caBIO domain objects

- Pathway image manipulation utility

CABIO ARCHITECTURE

This chapter describes the architecture of the caBIO system. It includes information about the major components, such as database object-relational mappings (ORM), client-server communication, and how the system connects to non-ORM sources, such as FreestyleLM. It also describes the layout of the caBIO system packages. Topics in this chapter include:

- *caBIO System Architecture* on this page

- *Client Technologies* on page 15

- *Major caBIO Domain Packages* on page 16

- *The caBIO System Packages* on page 17

caBIO System Architecture

The caBIO infrastructure exhibits an *n*-tiered architecture with client interfaces, server components, backend objects, data sources, and additional backend systems (*Figure 3.1*). This *n*-tiered system divides tasks or requests among different servers and data stores. This isolates the client from the details of where and how data is retrieved from different data stores. The system also performs common tasks such as logging and provides a level of security.

Clients (browsers, applications) receive information from backend objects. Java applications also communicate with backend objects via domain objects packaged within `cabio43-beans.jar`. Non-Java applications can communicate via SOAP (Simple Object Access Protocol). Back-end objects communicate directly with data sources, either relational databases (using Hibernate) or non-relational systems.
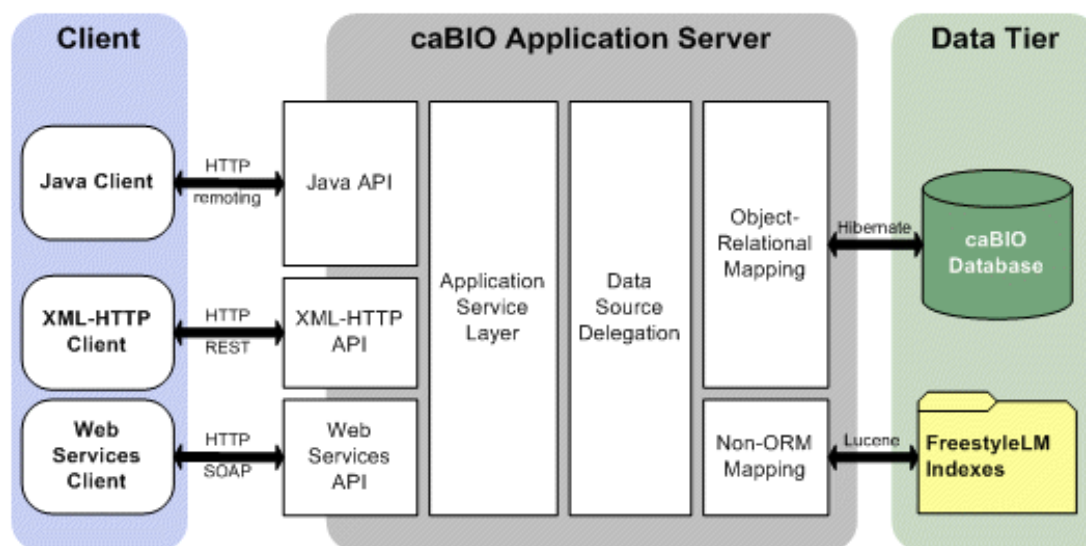
Figure 3.1 caBIO architecture

Most of the caBIO infrastructure is written in the Java programming language and leverages reusable, third-party components.

The infrastructure is composed of the following layers.

The Application Service Layer

The Application Service layer consolidates incoming requests from various interfaces and translates them to native query requests that are then passed to the data layers. This layer is also responsible for handling client authentication and access control using the Java API. (This feature is currently disabled for the caBIO system running at NCI CBIIT; all interfaces provide full, anonymous read-only access to all data.)

The Data Source Delegation Layer

This layer is responsible for conveying each query that it receives to the respective data source that can perform the query. The presence of this layer enables multiple data sources to be exposed by a single running instance of a caBIO server.

Object-Relational Mapping (ORM)

The ORM mapping is implemented using Hibernate. Hibernate is a high performance object/relational persistence and query service for Java. Hibernate provides the ability to develop persistent classes following common object-oriented (OO) design methodologies such as association, inheritance, polymorphism, and composition.

The Hibernate Query Language (HQL), designed as a "minimal" object-oriented extension to SQL, provides a bridge between the object and relational databases. Hibernate allows for real world modeling of biological entities without creating complete SQL-based queries to represent them.

HQL is fully object-oriented, understanding notions like inheritence, polymorphism and association. Detailed documentation is beyond the scope of this guide. See *Appendix A, References*, on page 97 for technical articles related to HQL.

Non-ORM Layer

The caBIO database is a relational database. A Lucene-based document index is generated on the backend to facilitate the Lucene full-text search feature. The caBIO API implements the non-ORM based Lucene search as well as the ORM based Hibernate Lucene Search.

Client Technologies

Applications using the Java programming language can access caBIO directly through the domain objects provided by the `cabio43-beans.jar` (see *caBIO System Architecture* on page 13.) The network details of the communication to the caCORE servers are abstracted away from the developer. Hence, developers need not deal with issues such as network and database communication, but can instead concentrate on the biological problem domain.

The caBIO system also allows non-Java applications to use SOAP clients to interface with caBIO web services. SOAP is a lightweight XML-based protocol for the exchange of information in a decentralized, distributed environment. It consists of an envelope that describes the message and a framework for message transport. caBIO uses the open source Apache Axis package to provide SOAP-based web services to users. This allows other languages, such as Python or Perl to communicate with caBIO objects in a straightforward manner.

The caBIO architecture includes a presentation layer that uses a J2SE application server (such as Tomcat or JBoss). The JSPs (Java Server Pages) are web pages with Java embedded in the HTML to incorporate dynamic content in the page. caBIO also employs Java Servlets, which are server-side Java programs that web servers can run to generate content in response to client requests. All logic implemented by the presentation layer uses Java Beans, which are reusable software components that work with Java. All caBIO domain objects can be transformed into XML, the eXtensible Markup Language, as a universal format for structured data on the Web.

Communication between the client interfaces and the server components occurs over the Internet using the HTTP protocol. The server components are deployed in a web application container as a .war (Web archive) file that communicates with the back-end relational database management system that contains the actual data.

By using XSL/XSLT, the extensible stylesheet language for expressing stylesheets and XSL Transformations (XSLT), as a language for transforming XML documents, non-programmers can transform the information in the caBIO objects for use in documents or other systems.

Major caBIO Domain Packages

Table 3.1 shows the major caBIO domain packages from which you can access the Java interfaces and classes. All of the objects in the domain packages implement the java.io.Serializable interface. To view the JavaDocs page for each package, go to http://cabioapi.nci.nih.gov/cabio43/docs.

Package	*Description*
caBIO	Contains the domain-specific classes in the caBIO object package such as Gene, Chromosome, etc. For a list of the domain objects and their descriptions, see *Chapter 5, Cancer Bioinformatics Infrastructure Objects,* on page 57. `gov.nih.nci.cabio.domain`
Common	Contains a single class, DatabaseCrossReference, which is used by the caBIO domain package objects to provide links to related data hosted by other sources. Further details about this class are available in the next section and in *Chapter 5, Cancer Bioinformatics Infrastructure Objects,* on page 57. `gov.nih.nci.common.domain`
Pathways	Contains classes supporting data from the Pathway Interaction Database (PID). Further details about the classes in this package are available in *Chapter 5, Cancer Bioinformatics Infrastructure Objects,* on page 57. `gov.nih.nci.cabio.pathways`
Provenance	Contains classes that provide provenance (source) metadata about the origin of data in caBIO objects. Further details about the classes in this package are available in *Chapter 5, Cancer Bioinformatics Infrastructure Objects,* on page 57. `gov.nih.nci.common.provenance.domain`

Table 3.1 Major caBIO domain packages

The caBIO System Packages

Figure 3.2 displays a package for each application area in the caBIO API, described in *Table 3.2*.
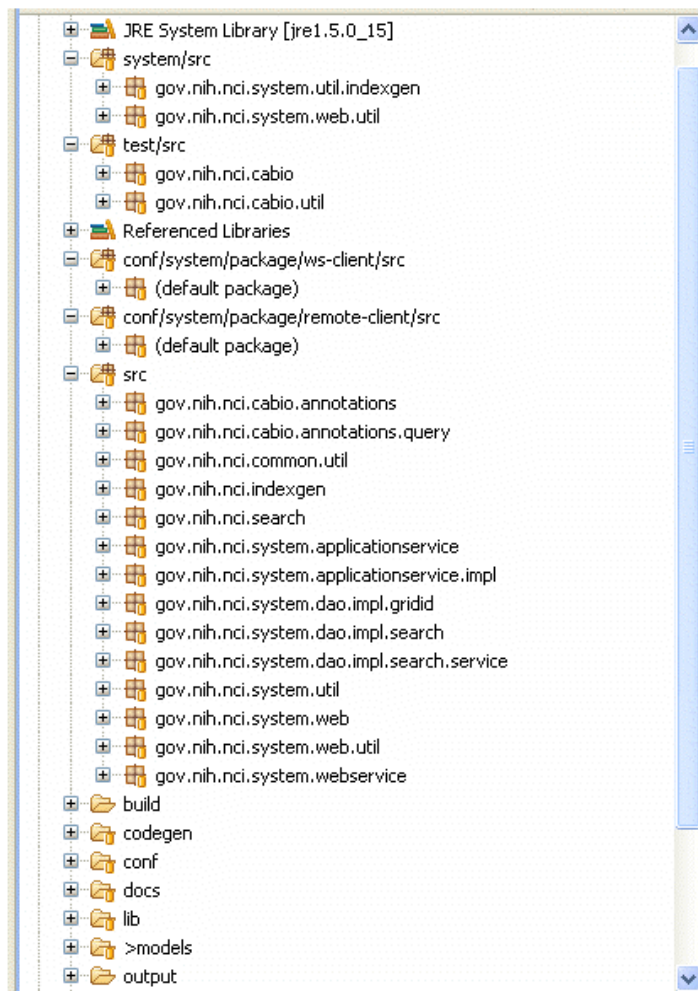

Figure 3.2 caBIO API package

In addition to domain packages, the caBIO client API includes the following framework packages:

Package	Description
`gov.nih.nci.common`	The common package contains utility classes for manipulating caBIO domain objects.
`gov.nih.nci.indexgen`	The indexgen package contains the classes that generate the Lucene indexes for the caBIO database.
`gov.nih.nci.search`	The search package contains domain objects for the FreestyleLM search component and Grid Ids. Clients can instantiate these objects and pass them to appropriate API methods to search the FreestyleLM indexes, Range Queries or the Grid Ids. For the range queries, it includes subclasses to search on a specific assembly of a genome as well as to search for locations relative to a genomic feature (defined by the subclass). It is possible to specify distances between, upstream and downstream of the feature.
`gov.nih.nci.system`	The system package has contains the client-side API, which communicates with the caBIO server, and helper classes for creating client-side proxies.
`gov.nih.nci.system.applicationService`	The ApplicationService package contains classes that extend the SDK-provided functionality to provide caBIO-specific functionalities such as searching for a caBIO domain objects by passing Grid Id, RangeQuery or FreestyleLM SearchQuery object.
`gov.nih.nci.system.client`	The client package consists of classes that are used to generate dynamic proxies. The Application Service proxy classes intercept all the calls to the actual ApplicationService and takes action to facilitate the lazy-loading mechanism for the caBIO API.
`gov.nih.nci.system.dao` (Data Access)	The data access package is the layer at which the query is executed. This layer has implementation for querying ORM and non-ORM based data stores and as with the applicationServices, provides functionality to perform Grid Id, Range Query and FreestyleLM searches
`gov.nih.nci.system.webservice`	The Web service package contains the Web service wrapper class that uses Apache's Axis and has been extended to provide caBIO-specific functionality like searching for domain objects using grid Ids.

Table 3.2 caBIO API framework client packages

INTERACTING WITH CABIO

caBIO provides facilities enabling both end user interaction via Graphical User Interfaces (GUIs) and system interaction via the caBIO Service Interface Layer. This chapter describes the various methods for interacting with caBIO and provides examples of how to use each of the interface APIs.

Topics in this chapter include:

Tip: Code snippets in this and other chapters of the technical guide PDF can be copied for pasting elsewhere. Right-click anywhere in the PDF and select **Allow Hand Tool to Select Text**. Click and drag to select text. Copy the text, then paste the text into NotePad or WordPad to retain the line distinctions.

Accessing caBIO

caBIO provides two primary interfaces for accessing data:

- caBIO Home Page: http://cabioapi.nci.nih.gov/cabio43/Home.action

- caBIO Portlet: http://cagrid-portal.nci.nih.gov/web/guest/community

caBIO Home Page

The caBIO home page (*Figure 4.1*) is the entry point for browsing and querying the caBIO object model using a Query-by-Example (QBE)-style interface. The page is auto-generated using the caCORE SDK and is customized to support unique caBIO features such as FreestyleLM (Freestyle Lexical Mine) search, a full text search (Google™-like search) capability. Users build queries using the attributes in the caBIO Service Interface Paradigm for each class (see *caBIO Service Interface Paradigm* on page 27 for more information).

Figure 4.1 caBIO Home Page

The FreestyleLM search component consists of interfaces and APIs for conducting horizontal searches across caBIO domain objects. For example, when searching for `brca1`, the FreestyleLM search returns any object that includes the `brca1` gene in its attributes, such as Genes, Proteins, and Pathways (see *Figure 4.2*).

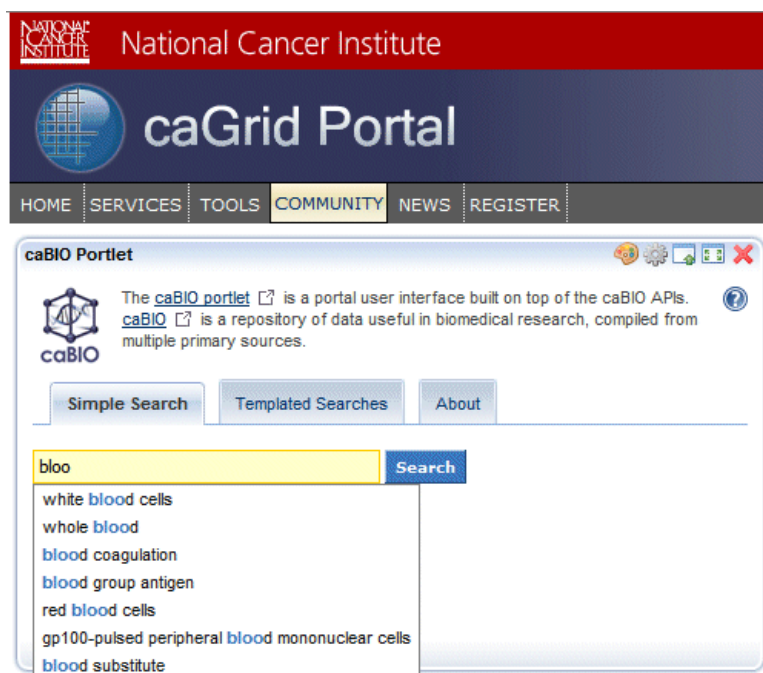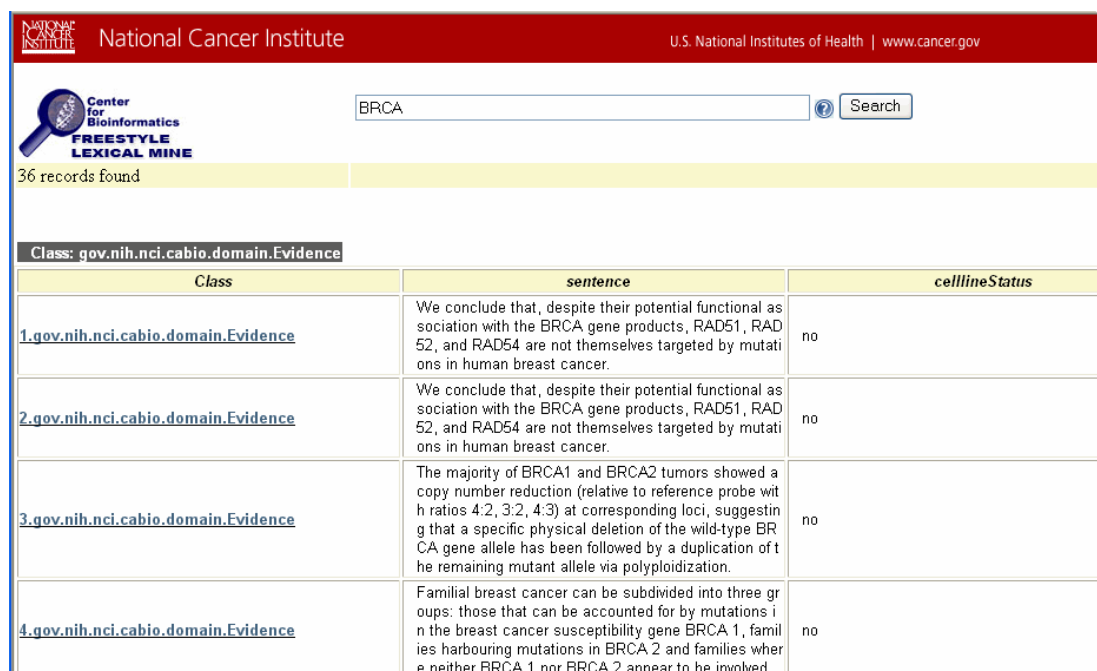*Figure 4.2 caBIO Portlet — Simple search auto-complete*

Figure 4.3 illustrates the results of a FreeStyleLM search.

Figure 4.3 FreestyleLM search results

Users can query caBIO via the QBE interface illustrated in *Figure 4.4*. Users can traverse the caBIO object model and enter search criteria for each object attribute

Figure 4.4 Query-by-Example (QBE) search interface

Search results for the QBE search appear in tabular form (*Figure 4.5*). Users can retrieve additional information from associated objects by clicking a relevant link.

Figure 4.5 QBE search results

caBIO Portlet

A portlet is a user interface component that operates within a web portal. The caBIO Portlet is a GUI component that operates within the caGrid Portal as a community service. The caBIO Portlet provides a simple search for performing "Google-like" queries and a templated-based search allowing users to execute pre-defined queries.

The simple search (*Figure 4.6*) is based on the existing FreestyleLM search that allows a user to search by entering one or more keywords. As the user types in the search box, the FreestyleLM search suggests keywords to search on. The results of the simple search are formatted based on the caBIO object type to show a relevant title and description and class for each result. Pagination allows the user to retrieve further results. All functionality in the simple search is implemented using Ajax with the REST API.

Figure 4.6 caBIO Portlet — Simple search

Results from Simple Searches are displayed in "Google-like" form (*Figure 4.7*). Additional details can be obtained for each search result by selecting the search result link.
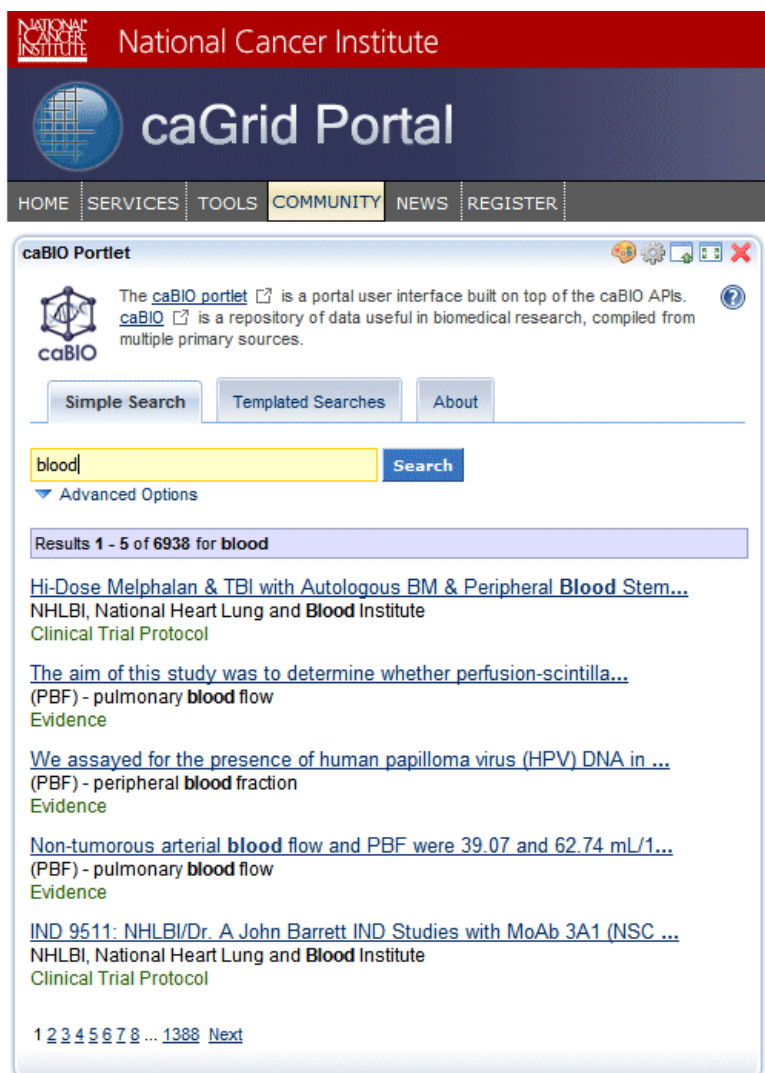
Figure 4.7 caBIO Portlet — Simple search results

Templated searches provide the user with pre-defined searches categorized by functionality.

Templated searches supported by caBIO include:

- Genome Range Queries – Absolute range queries by Taxon or Chromosome and Grid Id range queries

- Microarray Annotations – Reporter lookup by Gene, SNP, or Reporter Name

- Genomic Annotations – Gene lookup by Gene symbol or HUGO symbol and Gene Ontology Lookup by Protein (see *Figure 4.8* on page 25)

- Pathways – Pathway search by Pathway name, Pathway by Source/Provider, Pathway by Protein, or Gene symbol (see *Figure 4.9* and *Figure 4.10* on page 26)

- Cancer Gene Index – Search for Diseases or Agents associated with a Gene

Figure 4.8 caBIO Portlet — Genomic Annotations templated search example

Figure 4.9 caBIO Portlet — Pathway by Protein templated search example

Figure 4.10 caBIO Portlet — Pathway by Source templated search example

Results from templated searches are displayed in tabular form (see *Figure 4.11*). You can view additional details for each search result by clicking the **Details** link.

Figure 4.11 Templated search results

The caBIO Portlet is available on the Community tab of the caGrid portal, http://cagrid-portal.nci.nih.gov/web/guest/community.

caBIO Service Interface Paradigm

The caBIO API is built on top of the caCORE SDK. The caCORE architecture includes a service layer that provides a single, common access paradigm to clients using any of the provided interfaces. As an object-oriented middleware layer designed for flexible data access, caBIO relies heavily on strongly-typed objects and an object-in/object-out mechanism. The methodology used for obtaining data from the caBIO system is often referred to as query-by-example, meaning that the inputs to the query methods are themselves domain objects that provide the criteria for the returned data. The major benefit of this approach is that it allows for run-time semantic interoperability when used as part of the caCORE infrastructure, which provides shared vocabularies and a metadata registry.

The basic order of operations required to access and use the caBIO system is as follows:

1. Ensure that the client application has knowledge of the objects in the domain space.

2. Formulate the query criteria using the domain objects.

3. Establish a connection to the server.

4. Submit the query objects and specify the desired class of objects to be returned.

5. Use and manipulate the result set as desired.

There are five primary application programming interfaces (APIs) native to the caBIO system. Each of the available interfaces described below uses this same paradigm to provide access to the caBIO domain model, but with minor changes relating primarily to the syntax and structure of the clients. The following sections describe each API, identify installation and configuration requirements, and provide code examples.

Java API

The Java API provides direct access to domain objects and all service methods. Because caBIO is natively built in Java, this API provides the fullest set of features and capabilities.

Installation and Configuration

The caBIO Java API uses the following on the client machine (*Table 4.1*).

Software	Version	Required
Java 2 Platform Standard Edition Software 5.0 Development Kit (JDK 5.0)	1.5.0_14 or higher	Yes
Apache Ant	1.6.5 or higher	Yes

Table 4.1 caBIO Java API client software

Accessing the caBIO system requires an Internet connection.

To use the Java API, download the client package provided on the NCI CBIIT web site.

To install and configure the Java API, follow these steps:

1. Open your browser and go to http://ncicb.nci.nih.gov. (*Figure 4.12*)
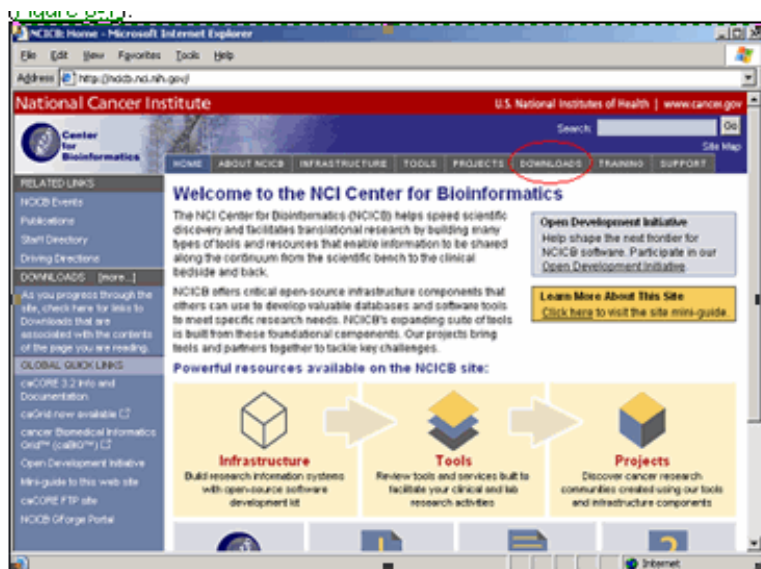
Figure 4.12 NCI CBIIT website downloads option

2. Click the **Download** link on the menu bar.

3. Scroll down to the section titled **caBIO** and click the **Download** link.

4. On the provided form, enter your name, email address and institution name and click to **Enter the Download Area**.

5. Accept the license agreement.

6. On the caBIO downloads page, download the caBIO Zip file from the **Primary Distribution** section.

7. Extract the contents of the downloadable archive to a directory on your hard drive (for example, `c:\cabio` on Windows or `/usr/local/cabio` on Linux). The extracted directories and files include those in *Table 4.2*

Directories and Files	*Description*	*Component*
`build.xml`	Ant build file	Build file
`Src directory`	Contains demo code	Sample code
`TestClient.java`	Java API client sample	SDK sample code
`TestGetXMLClient.java`	REST client sample	
`TestXMLClient.java`	XML marshalling sample	
`TechGuideExamples.java`	Java API examples from this guide	caBIO sample code
`TestQueries.java`	Java API query examples	
`TestFreestyleLM.java`	FreestyleLM search examples	
`TestCQL.java`	CQL API examples	
`TestSVG.java`	SVG pathway image examples	
`TestXML.java`	XML utility sample	
`lib directory`	contains required jar files	Java Libraries
`cabio43-beans.jar`	domain objects	caBIO API
`cabio43-client-framework.jar`	caBIO API extensions to the caCORE SDK Framework	
`Spring.jar`	Spring framework	HTTP Remoting
`hibernate*.jar`	Hibernate	ORM layer
`cglib-2.1.3.jar`	CGLIB proxy library	
`Log4j-1.2.13.jar`	Log4j and Commons Logging	Logging Framework
`commons-logging.jar`		
`commons-discovery-0.2.jar`		
`castor-1.0.2.jar`	Castor serializer/deserializer	XML conversion
`xercesImpl.jar`	Apache Xerces XML parser	
`*.xsd`	XML schemas for objects	

Table 4.2 Extracted directories and files in caBIO Java client package

29

Directories and Files	*Description*	*Component*
`handle.jar`	Handle Service	Handle Service
`handlerservice.jar`	Custom handle code	
`conf directory`	Configuration files	
`xml-mapping.xml`	XML serialization mapping configuration files	XML mapping
`unmarshaller-xml-mapping.xml`		
`application-config-client.xml`	Spring configuration for client	Java API Spring configuration
`Log4j.properties`	Logging utilities configuration properties	Logging configuration

Table 4.2 Extracted directories and files in caBIO Java client package (Continued)

All of the `jar` files provided in the `lib` directory of the caBIO client package in addition to the files in the conf directory are required to use the Java API. These should be included in the Java classpath when building applications. The `build.xml` file that is included demonstrates how to do this when using Ant for command-line builds. If you are using an integrated development environment (IDE) such as Eclipse, refer to the tool's documentation for information on how to set the classpath.

A Simple Example

To run the simple example program after installing the caBIO client, open a command prompt or terminal window from the directory where you extracted the downloaded archive and enter ant rundemo. This will compile and run the `TechGuideExamples` class; successfully running this example indicates that you have properly installed and configured the caCORE client. The following is a short segment of code from the `TechGuideExamples` class along with an explanation of its functioning.

```
1  ApplicationService appService = ApplicationServiceProvider.getApplicationService();
2  Gene gene = new Gene();
3  gene.setSymbol("brca*");  // searching for all genes whose symbols start with brca
4  try {
5     List resultList = appService.search(Gene.class, gene);
6     for (Iterator resultsIterator = resultList.iterator(); resultsIterator.hasNext();)
7     {
8         Gene returnedGene = (Gene) resultsIterator.next();
9         System.out.println("Symbol: " + returnedGene.getSymbol() +
10            "\tName " + returnedGene.getFullName()) +
11            "\tTaxon:" + returnedGene.getTaxon().getScientificName();
12        }
13 } catch (Exception e) {
14     e.printStackTrace();
15 }
```

This code snippet creates an instance of a class that implements the ApplicationService interface. This interface defines the service methods used to access data objects. A criterion object is then created that defines the attribute values for which to search. The search method of the ApplicationService implementation is called with parameters that indicate the type of objects to return, `Gene.class`, and the criteria that returned objects must meet, defined by the gene object. The search method returns objects in a List collection, which is iterated through to print some basic information about the objects.

Although this is a simple example of the use of the Java API, a similar sequence can be followed with more complex criteria to perform sophisticated manipulation of the data provided by caBIO. Additional information and examples are provided in the sections that follow.

Service Methods

The methods that provide programmatic access to the caBIO server are located in the `gov.nih.nci.system.applicationservice` package. The `ApplicationServiceProvider` class uses the factory design pattern to return an implementation of the ApplicationService interface. The provider class determines whether there is a locally running instance of the caBIO system or whether it should use a remote instance. The returned ApplicationService implementation exposes the service methods that enable read/write operations on the domain objects.

Note: Although the infrastructure is capable of write operations, this functionality has been disabled for caBIO because it is primarily meant as a read-only data system.

The separation of the service methods from the domain classes is an important architectural decision that insulates the domain object space from the underlying service framework. As a result, new business methods can be added without needing to update any of the domain model or the associated metadata information from the object model. (This is critical for ensuring semantic interoperability over multiple iterations of architectural changes.)

Within the `ApplicationService` implementation, a variety of methods are provided allowing users to query data based on the specific needs and types of queries to be performed. In general, there are five types of searches:

- **Simple searches** are those that take one or more objects from the domain models as inputs and return a collection of objects from the data repositories that meet the criteria specified by the input objects.

- **Nested searches** also take domain objects as inputs but determine the type of objects in the result set by traversing a known path of associations from the domain model.

- **Detached criteria searches** use Hibernate detached criteria objects to provide a greater level of control over the results of a search (such as boolean operations, ranges of values, etc.)

- **HQL searches** provide the ability to use the Hibernate Query Language for the greatest flexibility in forming search criteria.

- **SDK Query Object criteria searches** were modeled similar to the Object representation of caBIG Query Language (CQL). The SDK Query Object criteria searches use syntax similar to the query by example (QBE) to specify the way results are to be retrieved. The system formulates the query based on

the navigation path specified in the query search criteria. The query mechanism allows the user to search for the objects in platform independent query syntax.

Method Signature	`List search(`
	 `    Class targetClass,`
	 `    Object obj)`

Search Type	Simple (One criteria object)
Description	Returns a List collection containing objects of type `targetClass` that conform to the criteria defined by `obj`
Example	`search(Gene.class, gene)`

Method Signature	`List search(`
	 `    Class targetClass,`
	 `    Object objList)`

Search Type	Simple (One criteria object)
Description	Returns a List collection containing objects of type `targetClass` that conform to the criteria defined by a collection of objects in `objList`. The returned objects must meet ANY criteria in `objList` (i.e. a logical OR is performed).
Example	`search(Gene.class, geneCollection)`

Method Signature	`List search(`
	 `    String path,`
	 `    Object obj)`

Search Type	Nested
Description	Returns a List collection containing objects conforming to the criteria defined by obj and whose resulting objects are of the type reached by traversing the node graph specified by path
Example	`search("gov.nih.nci.cabio.domain.Protein,` `gov.nih.nci.cabio.domain.Gene", nucleicAcidSequence)`

Method Signature	`List search(`
	 `    String path,`
	 `    List objList)`

Search Type	Nested
Description	Returns a List collection containing objects conforming to the criteria defined by the objects in `objList` and whose resulting objects are of the type reached by traversing the node graph specified by path
Example	`search("gov.nih.nci.cabio.domain.Protein,` `gov.nih.nci.cabio.domain.Gene", sequenceList)`

Method Signature	List query(DetachedCriteria detachedCriteria)
Search Type	Detached criteria
Description	Returns a List collection conforming to the criteria specified by `detachedCriteria`.
Example	`query(criteria)`

Method Signature	List query(
	        Object criteria,
	         Integer firstRow,
	        String targetClassName)

Search Type	Detached criteria
Description	Identical to the previous query method, but allows for control over the size of the result set by specifying the row number of the first row.
Example	`query(criteria, 101, targetClassName)`

Method Signature	List query(HQLCriteria hqlCriteria)
Search Type	Hibernate Query Language
Description	Returns a List collection of objects that conform to the query in HQL syntax contained in `hqlCriteria`.
Example	`query(hqlCriteria)`

Method Signature	List query(CQLQuery cqlQuery)
Search Type	Common Query Language
Description	Returns a List collection of objects that conform to the query in SDK Query Object syntax contained in `cqlQuery`.
Example	`query(cqlQuery)`

In addition to the data access methods, the following helper methods are available via the ApplicationService class that provide flexibility in controlling queries and result sets.

Method Signature	int getQueryRowCount(
	    Object criteria,
	    String targetClassName

Description	Gets the number of objects in a given query. This is useful in determining the size of the data before querying for data
Example	`getQueryRowCount(criteria,` `"gov.nih.nci.cabio.domain.Gene")`

Examples of Use

This section includes a number of examples that demonstrate the use of the caBIO APIs. Included with each example is a brief description of the type of search being performed, a UML diagram depicting the domain objects used, and the example code accompanied by explanatory text.

Example One: Simple Search (Single Criteria Object)

In this example, a search is performed for all genes whose symbols start with 'brca'. The code iterates through the returned objects and prints out the symbol and name of each object along with the name of an associated object of type Taxon. The fetch of the associated Taxon object is done in the background and is completely transparent to the ser.
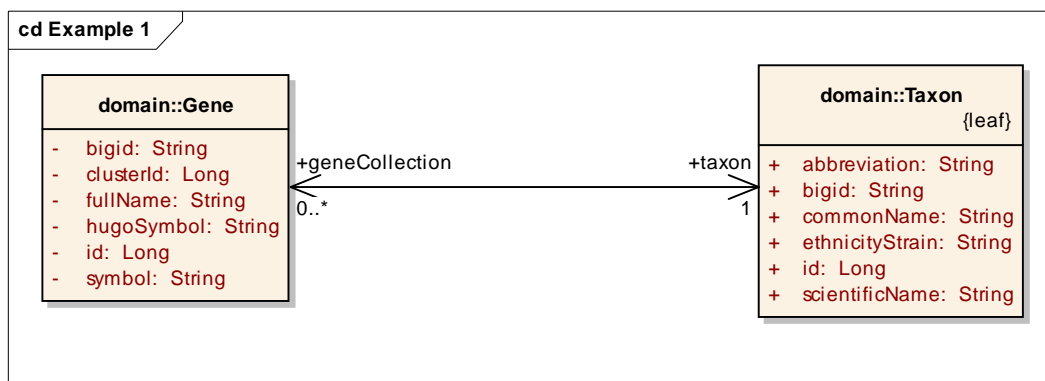

```
1   ApplicationService appService = ApplicationServiceProvider.getApplicationService();
2   Gene gene = new Gene();
3   gene.setSymbol("brca*");  // searching for all genes whose symbols start with 'brca'
4   try
5   {
6      List resultList = appService.search(Gene.class, gene);
7      for (Iterator resultsIterator = resultList.iterator(); resultsIterator.hasNext();)
8      {
9         Gene returnedGene = (Gene) resultsIterator.next();
10        System.out.println("Symbol: " + returnedGene.getSymbol() +
12           "\tTaxon:" + returnedGene.getTaxon().getScientificName()
11           "\tName " + returnedGene.getFullName());
13     }
14  } catch (Exception e) {
15       e.printStackTrace();
16  }
```

Lines	Description
1	Creates an instance of a class that implements the ApplicationService interface; this interface defines the service methods used to access data objects.
2-3	Creates a criterion object that defines the attribute values for which to query.
6	Calls the search method of the ApplicationService implementation and passes it the type of objects to return, Gene.class, and the criteria that returned objects must meet, defined by the gene object; the search method returns objects in a List collection.

Lines	Description
9	Casts an object from the result List and creates a variable reference to it of type Gene.
10	Prints the symbol attribute.
11	Prints the fullName attribute.
12	Fetches an associated object of type Taxon and prints its scientificName attribute.

Example Two: Simple Search (Criteria Object Collection)

This example demonstrates a search with multiple criteria objects that are passed to the search() method. The result set includes all objects of the specified type that match ANY of the criteria objects. In this case, the search returns all objects of type Gene that are associated with Taxon objects whose abbreviation attribute is either "hs" or "ms". In biological terms, this search returns all human and mouse genes.

```
1   Taxon taxon1 = new Taxon();
2   taxon1.setAbbreviation("Hs");          // Homo sapiens
3   Taxon taxon2 = new Taxon();
4   taxon2.setAbbreviation("Mm");          // Mus musculus
5   List<Taxon> taxonList = new ArrayList<Taxon>();
6   taxonList.add(taxon1);
7   taxonList.add(taxon2);
8   try
9   {
10      List resultList = appService.search(Gene.class, taxonList);
11      System.out.println("Number of results: "+resultList.size());
12  }
13  catch (Exception e) {
14      e.printStackTrace();
15  }
```

Lines	Description
1-4	Creates two Taxon objects describing the search criteria.
5-7	Creates a List collection containing the two Taxon objects.
10	Searches for all Gene objects where the associated Taxon object matches ANY of the objects found in the taxonList collection.

Example Three: Simple Search (Compound Criteria Object)

In this example, the object that is passed to the search() method contains criteria values that are found in associated objects and collections of objects. This query returns those objects that match all of the criteria in the compound object. Note the distinction between this type of search and the previous example in which a collection of objects is passed into the search method. In the last example, the results will match ANY of the criteria objects. In this example, however, where a single compound object

is used, ALL criteria are matched. In biological terms, this search will return all pathways associated with the human Interleukin 5 gene.
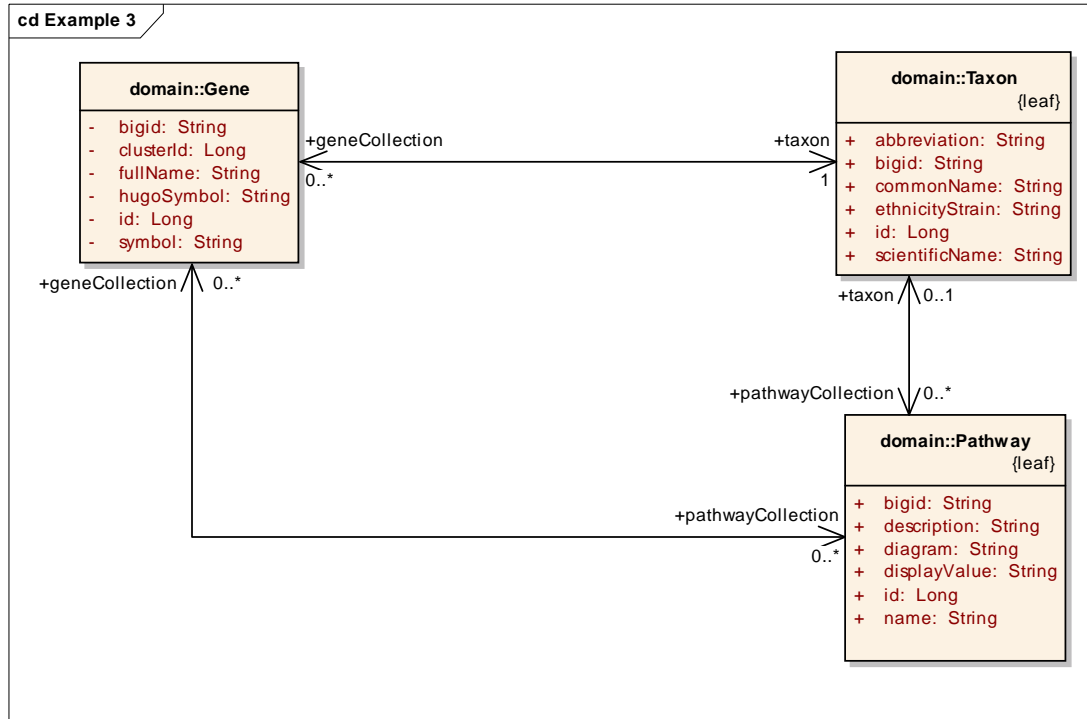

```
1   Taxon taxon = new Taxon();
2   taxon.setAbbreviation("hs");          // Homo sapiens
3   Gene gene = new Gene();
4   gene.setTaxon(taxon);
5   gene.setSymbol("IL5");                // Interleukin 5
6   List<Gene> geneList = new ArrayList<Gene>();
7   geneList.add(gene);
8   Pathway pathway = new Pathway();
9   pathway.setGeneCollection(geneList);
10  try
11  {
12     List resultList = appService.search("gov.nih.nci.cabio.domain.Pathway", pathway);
13     for (Iterator resultsIterator = resultList.iterator(); resultsIterator.hasNext();)
14     {
15        Pathway returnedPathway = (Pathway)resultsIterator.next();
16        System.out.println("Name: "+returnedPathway.getName()
17           + "\tDisplayValue: " + returnedPathway.getDisplayValue());
18     }
19  } catch (Exception e) {
20     e.printStackTrace();
21  }
```

Lines	Description
1-5	Creates a Gene object and sets the symbol to "IL5" and the associated Taxon to an object whose abbreviation is set to "hs".
6-7	Because the Pathway and Gene classes are related by a many-to-many association, it is necessary to create a collection to contain the Gene object that will act as part of the compound criteria; multiple Gene objects could be added to this collection as needed.
8-9	Creates a Pathway object and sets the value of its geneCollection to the geneList object just created.

Lines	Description
12	Searches for all Pathway objects whose geneCollection contains objects that match the set criteria (i.e. the symbol is "IL5" and the associated Taxon objects' abbreviations are set to "hs").

Example Four: Nested Search

A nested search is one where a traversal of more than one class-class association is required to obtain a set of result objects given the criteria object. This example demonstrates one such search in which the criteria object passed to the search method is of type Gene, and the desired objects are of type ProteinSequence. Because there is no direct association between these two classes, the path of the traversal is passed to the search method enabling the query to be performed.
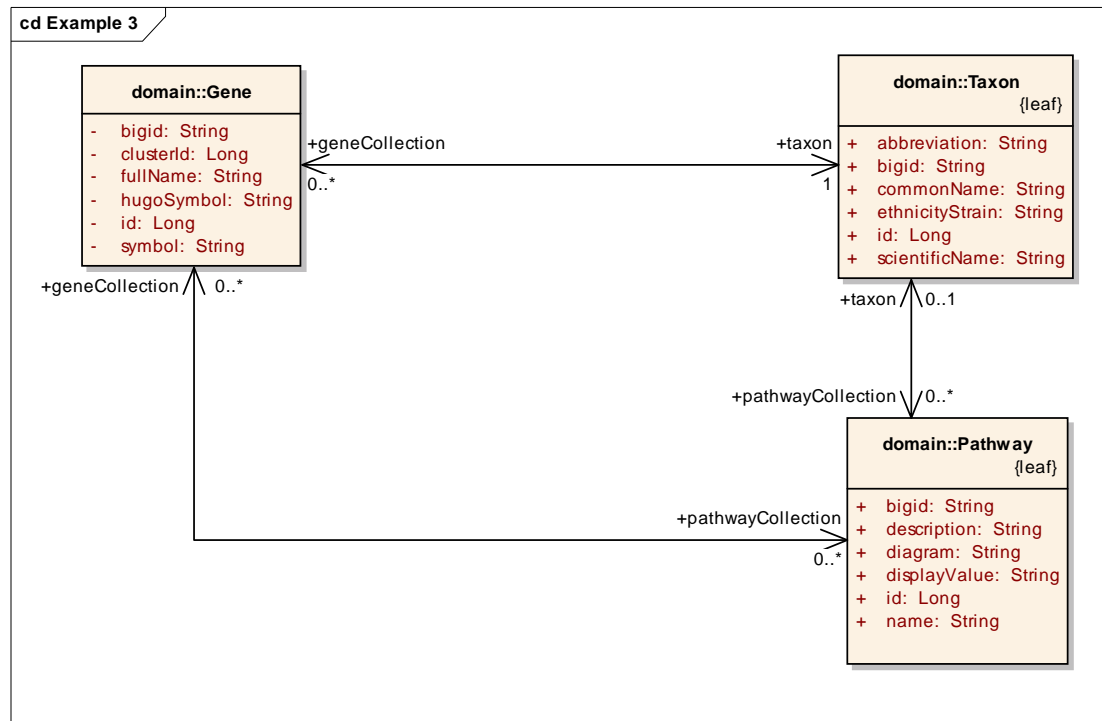

```
1   Gene gene = new Gene();
2   gene.setSymbol("TP53");              // Tumor protein p53 (Li-Fraumeni syndrome)
3   try
4   {
5      List resultList = appService.search(
6         "gov.nih.nci.cabio.domain.ProteinSequence,gov.nih.nci.cabio.domain.Protein"
7         gene);
8      for (Iterator resultsIterator = resultList.iterator(); resultsIterator.hasNext
9      {
10        ProteinSequence returnedProtSeq = (ProteinSequence)resultsIterator.next();
11        System.out.println("Id: " + returnedProtSeq.getId() +
12                           "Length: " + returnedProtSeq.getLength() );
13     }
14  } catch (Exception e) {
15     e.printStackTrace();
16  }
```

Lines	Description
1-2	Creates a Gene object and sets the symbol to "TP53".
6	Defines search path as traversing from the criteria object of type Gene through Protein to ProteinSequence; note that the first element in the path is the desired class of objects to be returned, and that subsequent elements traverse back to the criteria object.
7	Sets the criteria object to the previously-created Gene.

Example Five: Detached Criteria Search

This example demonstrates the use of Hibernate detached criteria objects to formulate and perform more sophisticated searches. A detailed description of detached criteria is beyond the scope of this document; for more information, please consult the Hibernate documentation at http://www.hibernate.org/hib_docs/v3/api/org/hibernate/criterion/DetachedCriteria.html.

cd Example 5

Location

domain::PhysicalLocation

- assembly: String
- chromosomalEndPosition: Long
- chromosomalStartPosition: Long
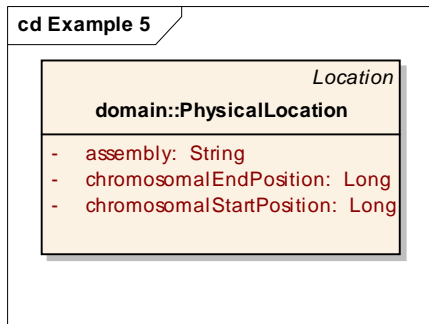
```
 1 DetachedCriteria criteria = DetachedCriteria.forClass(PhysicalLocation.class);
 2 try {
 4     DetachedCriteria criteria = DetachedCriteria.forClass(PhysicalLocation.class);
 5     criteria.add(Restrictions.eq("assembly", "reference"));
 6     criteria.add(Restrictions.gt("chromosomalStartPosition", new Long(86851632)));
 7     criteria.add(Restrictions.lt("chromosomalEndPosition",  new Long(86861632)));
 8     List resultList = appService.query(criteria);
 9     System.out.println("\n Total # of  records = " + resultList.size());
10 }
11 catch (Exception e) {
12     e.printStackTrace();
13 }
```

Lines	Description
1	Creates a DetachedCriteria object and sets the class on which the criteria will operate to PhysicalLocation.
5	Sets a restriction on the objects which states that the attribute assembly must be equal to "reference".
6	Sets a restriction on the objects that states that the attribute chromosomalStartPosition must be greater than ("gt") the value 86851632.
7	Sets a restriction on the objects that states that the attribute chromosomalEndPosition must be less than ("lt") the value 86861632.
8	Calls the query method of the ApplicationService implementation, passing the detached criteria object.

Example Six: HQL Search

In this example, a search is performed for all genes whose symbols start with BRCA. This is identical to Example One but uses a Hibernate Query Language (HQL) search string to form the query. For more information on HQL syntax, consult the Hibernate documentation at http://www.hibernate.org/hib_docs/v3/reference/en/html/queryhql.html.

```
1 try {
2   String hqlString = "FROM gov.nih.nci.cabio.domain.Gene g
      WHERE g.symbol LIKE ?";
3     List<String> params = new ArrayList<String>();
4     params.add("BRCA%");
5     HQLCriteria hqlC = new HQLCriteria(hqlString, params);
6     List resultList = appService.query(hqlC);
7     System.out.println("\n Total # of records = " +
      resultList.size());
8 }
9 catch (Exception e) {
10    e.printStackTrace();
11 }
```

Example Seven: SDK Query Object Search

SDK Query Object Example One

```
1       ApplicationService appService = ApplicationServiceProvider.getApplicationService();

2       CQLQuery cqlQuery = new CQLQuery();
3       CQLObject target = new CQLObject();  // Create Gene Object
4       target.setName("gov.nih.nci.cabio.domain.Gene");

5       CQLAttribute attribute = new CQLAttribute();
6       attribute.setName("symbol");
7       attribute.setValue("%il%");
8       attribute.setPredicate(CQLPredicate.LIKE);
9       target.setAttribute(attribute);

10      cqlQuery.setTarget(target);

11      try {
12          List resultList = appService.query(cqlQuery, gov.nih.nci.cabio.domain.Gene");
14          //Iterate through retrieved list of objects
15          for (Iterator i = resultList.iterator(); i.hasNext();) {
17              Gene returnedGene = (Gene) i.next();
18              System.out.println(
19                  "Symbol: " + returnedGene.getSymbol() + "\n" +
20                  "\tTaxon:" + returnedGene.getTaxon().getScientificName() + "\n" +
22                  "\tName " + returnedGene.getFullName() + "\n");
24          }
25      }
26      catch (Exception e) {
27              e.printStackTrace();
28      }
```

Lines	Description
2	Creates a new CQLQuery Object.
3-4	Creates a new search Target object (Gene).
5-8	Creates a new attribute for the search target object whose value should be compared with the value in the database for given predicate.
10	Sets the search target object in SDK Query Object Query.
12-13	Sets the search criteria object to the previously-created CQLQuery.

SDK Query Object Example Two

```
1       CQLQuery cqlQuery = new CQLQuery();
2       CQLObject target = new CQLObject();
3        target.setName("gov.nih.nci.cabio.domain.Gene");

4       CQLAttribute attribute = new CQLAttribute();
5       attribute.setName("symbol");
6       attribute.setValue("brca%");
7       attribute.setPredicate(CQLPredicate.LIKE);

8       //Create Taxon Object Query
9       CQLAssociation association1 = new CQLAssociation();
10      association1.setName("gov.nih.nci.cabio.domain.Taxon");
11      CQLAttribute attribute1 = new CQLAttribute();
12      attribute1.setName("abbreviation");
13      attribute1.setValue("%hs%");
14      attribute1.setPredicate(CQLPredicate.LIKE);
15      association1.setAttribute(attribute1);

16      //Create Taxon Object Query
17      CQLAssociation association2 = new CQLAssociation();
18      association2.setName("gov.nih.nci.cabio.domain.Taxon");
19      CQLAttribute attribute2 = new CQLAttribute();
20      attribute2.setName("abbreviation");
21      attribute2.setValue("%m%");
22      attribute2.setPredicate(CQLPredicate.LIKE);
23      association2.setAttribute(attribute2);

24      //Create Group(Collection) of Taxon Object Query
25      CQLGroup group = new CQLGroup();
26      group.addAssociation(association1);
27      group.addAssociation(association2);
28      group.setLogicOperator(CQLLogicalOperator.OR);
29      target.setAttribute(attribute);
30      target.setGroup(group);

31      cqlQuery.setTarget(target);

32      try {
33          List resultList = appService.query(cqlQuery);
34
35          //Iterate through retrieved list of objects
36          for (Iterator resultsIterator = resultList.iterator();
resultsIterator.hasNext();) {
37              Gene returnedGene = (Gene) resultsIterator.next();
38              System.out.println(
39                  "Symbol: " + returnedGene.getSymbol() + "\n" +
40                  "\tTaxon:" + returnedGene.getTaxon().getScientificName() +
41                  "\n" +"\tName " + returnedGene.getFullName() +"\n");
42          }
43      } catch (Exception e) {
44          e.printStackTrace();
45      }
```

Lines	Description
1	Creates a new CQLQuery Object.
2-3	Creates a new search Target object (Gene).
4-7	Sets attribute criteria for Target object
9-15, 17-23	Creates a new associated object (Taxon) whose attribute value should be compared with the value in the database for a given predicate.
25-30	Creates a group (collection) of objects and associates it to the search target object.
31	Sets the search target object in SDK Query Object Query.
33	Search using the previously-created CQLQuery.

SDK Query Object Example Three

```
1      ApplicationService appService =
                    ApplicationServiceProvider.getApplicationService();

2      CQLQuery cqlQuery = new CQLQuery();
3      CQLObject target = new CQLObject();
4      target.setName("gov.nih.nci.cabio.domain.Pathway");

5      //Create Gene Object Query
6      CQLAssociation association1 = new CQLAssociation();
7      association1.setName("gov.nih.nci.cabio.domain.Gene");
8      CQLAttribute attribute1 = new CQLAttribute();
9      attribute1.setName("symbol");
10     attribute1.setValue("IL5");
11     attribute1.setPredicate(CQLPredicate.EQUAL_TO);
12     association1.setAttribute(attribute1);

13     //Create Taxon Object Query
14     CQLAssociation association2 = new CQLAssociation();
15     association2.setName("gov.nih.nci.cabio.domain.Taxon");
16     CQLAttribute attribute2 = new CQLAttribute();
17     attribute2.setName("abbreviation");
18     attribute2.setValue("%hs%");
19     attribute2.setPredicate(CQLPredicate.LIKE);
20     association2.setAttribute(attribute2);

21     //Set Relatonship between Gene and Taxon
22     association1.setAssociation(association2);

23     //Set Relatonship between Pathway and Gene
24     //Role name is required as it can not be determined using
25     //reflection
26     association1.setTargetRoleName("geneCollection");
27     target.setAssociation(association1);

28     cqlQuery.setTarget(target);

29     try
30     {
31             List resultList = appService.query(cqlQuery);
32             //Iterate through retrieved list of objects
33             for (Iterator resultsIterator = resultList.iterator();
34                     resultsIterator.hasNext();)
35             {
36                     Pathway returnedPathway = (Pathway)resultsIterator.next();
37                     System.out.println(returnedPathway.getDisplayValue());
38             }
39     } catch (Exception e) {
40             e.printStackTrace();
41     }
```

Lines	Description
2	Creates a new CQLQuery Object.
3-4	Creates a new search Target object (Pathway).
5-12	Creates a new associated object (Gene) whose attribute value should be compared with the value in the database for a given predicate.
13-20	Creates a new associated object (Taxon) whose attribute value should be compared with the value in the database for a given predicate.
22	Creates a relationship between two objects (Gene and Taxon).
26-27	Creates a relationship between a target object (Pathway) and its associated object (Gene). The role name is the name of the association used to retrieve the associated object. It is required in case it can not be determined by reflection.
28	Sets the search target object in a SDK Query Object Query.
31	Sets the search criteria object to the previously-created CQLQuery.

SDK Query Object Example Four

```
1       ApplicationService appService =
                     ApplicationServiceProvider.getApplicationService();

2       CQLQuery cqlQuery = new CQLQuery();
3       CQLObject target = new CQLObject();
4       target.setName("gov.nih.nci.cabio.domain.ProteinSequence");

5       //Create Gene Object Query
6       CQLAssociation association1 = new CQLAssociation();
7       association1.setName("gov.nih.nci.cabio.domain.Gene");
8       CQLAttribute attribute1 = new CQLAttribute();
9       attribute1.setName("symbol");
10      attribute1.setValue("TP53");
11      attribute1.setPredicate(CQLPredicate.EQUAL_TO);
12      association1.setAttribute(attribute1);

13      //Create Protein Object Query
14      CQLAssociation association2 = new CQLAssociation();
15      association2.setName("gov.nih.nci.cabio.domain.Protein");

16      //Set Relatonship between Gene and Protein
17      //Role name is required as it can not be determined using
18      //reflection
19      association1.setTargetRoleName("geneCollection");
20      association2.setAssociation(association1);

21      //Set Relatonship between Protein and ProteinSequence
22      //Example of using relationship from target to source
24      association1.setSourceRoleName("proteinSequence");
25      target.setAssociation(association2);

26      cqlQuery.setTarget(target);

27      try
28      {
29          List resultList = appService.query(cqlQuery);
30          //Iterate through retrieved list of objects
31          for (Iterator resultsIterator = resultList.iterator();
32                  resultsIterator.hasNext();)
33          {
34              ProteinSequence returnedProtSeq =
35                  (ProteinSequence)resultsIterator.next();
36              System.out.println("ID: " + returnedProtSeq.getId() +
37                  "Length: " + returnedProtSeq.getLength());
38          }
39      } catch (Exception e) {
40              e.printStackTrace();
41      }
```

Lines	Description
2	Creates a new CQLQuery Object.
3-4	Creates a new search Target object (ProteinSequence).
5-12	Creates a new associated object (Gene) whose attribute value should be compared with the value in the database for a given predicate.
14-15	Creates a new associated object (Protein).
19-20	Creates a relationship between a source object (Protein) and its associated object (Gene). The role name is the name of the association used to retrieve the associated object. It is required in case it can not be determined by reflection.
24-25	Creates a relationship between a target object (ProteinSequence) and its associated object (Protein).

Lines	Description
26	Sets the search target object in a SDK Query Object Query.
29	Sets the search criteria object to the previously-created CQLQuery.

Utility Methods

XML Utility

The caBIO Java Client provides a convenience class called `XMLUtility` in the `gov.nih.nci.common.util` package that provides a short cut to the caCORE SDK's utility for converting domain objects between native Java objects and XML serializations based on standard XML schemas. The XML schemas for all domain objects in caBIO, directly generated from the UML model, are included in the downloadable archive. Currently, the XML generated using the `XMLUtility` class includes only the object attributes; associated objects are not included.

The configuration used by the XML utility is stored in two files, `xml-mapping.xml` and `unmarshaller-xml-mapping.xml`. These files define the binding between class, attribute names and the corresponding XML element and attribute names.

A default marshaller and unmarshaller are automatically registered by the caBIO convenience class; Developers wishing to use their own should bypass the caBIO class and instantiate the SDK's XMLUtility in the `gov.nih.nci.system.client.util.xml` package.

In the following code, the XML utility is used to serialize an object and save it to a file stream. A new object is then instantiated from the file using the utility.

```
1   // Assume an object of type Gene called myGene
2   File myFile = new File("myGene.xml");
3   FileWriter myWriter = new FileWriter(myFile);
4   XMLUtility myUtil = new XMLUtility();
5   myUtil.toXML(myGene,myWriter);
6   Gene myNewGene = (Gene)myUtil.fromXML(myFile);
7   bool isSameGene = myNewGene.equals(myGene);        // true
```

Lines	Description
2-3	Creates a new file stream where the XML serialization will be saved.
4	Creates a new XMLUtility object; in this case, the default marshaller, and unmarshaller will be used.
5	Serializes the myGene object to XML using the mapping file and writes the output to the file stream myGene.xml.
6	Creates a new object called myNewGene by invoking the fromXML() method of the XMLUtility class and casting it to the proper type.
7	The newly created Gene object is equivalent to the old one.

For additional details, consult the caBIO JavaDocs at http://cabioapi.nci.nih.gov/cabio43/docs/.

Web Services API

The caBIO Web services API allows access to caBIO data from development environments where the Java API cannot be used, or where use of XML Web services is more desirable. This includes non-Java platforms and languages such as Perl, C/C++, .NET Framework (C#, VB.Net), Python, etc.

The Web services interface can be used in any language-specific application that provides a mechanism for consuming XML Web services based on the Simple Object Access Protocol (SOAP). In these environments, connecting to caBIO can be as simple as providing the endpoint URL. Some platforms and languages require additional client-side code to handle the implementation of the SOAP envelope and the resolution of SOAP types. A list of packages catering to different programming languages is available at http://www.w3.org/TR/SOAP/ and at http://www.soapware.org/.

To maximize standards-based interoperability, the caBIO Web service conforms to the Web Services Interoperability Organization (WS-I) Basic Profile. The WS-I Basic Profile provides a set of non-proprietary specifications and implementation guidelines enabling interoperability between diverse systems. More information about WS-I compliance is available at http://www.ws-i.org.

On the server side, Apache Axis is used to provide SOAP-based inter-application communication. Axis provides the appropriate serialization and deserialization methods for the Java beans to achieve an application-independent interface. For more information about Axis, visit http://ws.apache.org/axis/.

Configuration

The caBIO WSDL file is located at http://cabioapi.nci.nih.gov/cabio43/services/caBIOService?wsdl.

In addition to describing the protocols, ports and operations exposed by the caBIO Web service, this file can be used by a number of IDEs and tools to generate stubs for caBIO objects. This allows code on different platforms to instantiate objects native to each for use as parameters and return values for the Web service methods. Consult the specific documentation for your platform for more information on how to use the WSDL file to generate class stubs.

The caBIO Web services interface has a single endpoint called caBIOService, which is located at http://cabioapi.nci.nih.gov/cabio43/services/caBIOService. Client applications should use this URL to invoke Web service methods.

Java WS Client

The Java WS client is a separate download from the Java API described earlier. For installation instructions, see the Java API on page 28. This distribution bundles the necessary Java libraries and example code that shows how to use the caBIO Web Service API.

Directories and Files	Description	Component
`build.xml`	Ant build file	Build
`build.properties`	Web Service configuration	Build
`src directory`	Contains demo code	Sample code

Directories and Files	*Description*	*Component*
`TestClient.java`	SDK WS client sample	
`TestWS.java`	Web services Java sample	
`lib directory`	contains required jar files	Java Libraries
`cabio43-beans.jar`	domain objects	caBIO API
`hibernate-search.jar`	Hibernate Search Annotations	ORM layer
`axis.jar`	Apache Axis	Web Services
`saaj.jar`	SOAP API for Java	
`jaxrpc.jar`	Java API for XML-based RPC	
`wsdl4j-1.5.1.jar`	WSDL for Java	
`commons-logging.jar`	Log4j and Commons Logging	Logging Framework
`commons-discovery-0.2.jar`		
`conf directory`	Configuration files	
`undeploy.wsdd`	XML serialization mapping configuration files	XML mapping
`application-config-client.xml`	Spring configuration for client	Java API Spring configuration
`Log4j.properties`	Logging utilities configuration properties	Logging configuration

All of the jar files provided in the lib directory of the caBIO client package in addition to the files in the `conf` directory are required to use the Java API. These should be included in the Java classpath when building applications. The `build.xml` file that is included demonstrates how to do this when using Ant for command-line builds. If you are using an integrated development environment (IDE) such as Eclipse, refer to the tool's documentation for information on how to set the classpath.

Operations

Through the caBIOService endpoint, developers have access to four operations:

Operation	`queryObject`

Input Schema	`<complexType>`
	`    <sequence>`
	`            <element name="in0"`
	`type="xsd:string"/>`
	`        <element name="in1"`
	`type="xsd:anyType"/>`
	`         </sequence>`
	`</complexType>`

| Output Schema | ```
<sequence>
        <element name="queryReturn"type=
"ArrayOf_xsd_anyType"/>
</sequence>
``` |
|---|---|
| **Description** | Performs a search for objects conforming to the criteria defined by input parameter in1 and whose resulting objects are of the type reached by traversing the node graph specified by parameter `in0`; the result is a set of serialized objects (the type `ArrayOf_xsd_anyType` resolves to a sequence of `xsd:anyType` elements) |

Operation	`Query`
| **Input Schema** | ```
<complexType>
    <sequence>
        <element name="in0" type="xsd:string"/>
        <element name="in1" type="xsd:anyType"/>
        <element name="in2" type="xsd:int"/>
        <element name="in3" type="xsd:int"/>
    </sequence>
</complexType>
``` |
| **Output Schema** | ```
<sequence>
   <element name="queryReturn"
type="ArrayOf_xsd_anyType"/>
</sequence>
``` |
| **Description** | Identical to the previous queryObject method, but allows for control over the result set by specifying the row number of the first row (in2) and the maximum number of objects to return (in3) |

Developers should be aware of a significant behavioral decision that has been made regarding the Web services interface. When a query is performed with this interface, returned objects do not contain or refer to their associated objects. This means that a separate query invocation must be performed for each set of associated objects that need to be retrieved. One of the examples below demonstrates this functionality.

Examples of Use

Example One: Simple Search

The following code demonstrates a simple query written in the Java language that uses the Web Services API. This example uses Apache Axis on the client side to handle the type mapping, SOAP encoding, and operation invocation.

```
1   Service  service = new Service();
2   Call call = (Call) service.createCall();
3
4   QName qnGene = new QName("urn:domain.cabio.nci.nih.gov", "Gene");
5   call.registerTypeMapping(
6      Gene.class,
7      qnGene,
8      new org.apache.axis.encoding.ser.BeanSerializerFactory(Gene.class, qnGene),
9      new org.apache.axis.encoding.ser.BeanDeserializerFactory(Gene.class, qnGene)
10     );
11
12  String url = "http://cabioapi.nci.nih.gov/cabio40/services/caBIOService";
13
14  call.setTargetEndpointAddress(new java.net.URL(url));
15  call.setOperationName(new QName("caBIOService", "queryObject"));
16  call.addParameter("arg1", org.apache.axis.encoding.XMLType.XSD_STRING, ParameterMode.IN);
17  call.addParameter("arg2", org.apache.axis.encoding.XMLType.XSD_ANYTYPE, ParameterMode.IN);
18  call.setReturnType(org.apache.axis.encoding.XMLType.SOAP_ARRAY);
19
20  gov.nih.nci.cabio.domain.Gene gene = new gov.nih.nci.cabio.domain.Gene();
21  gene.setSymbol("IL*");
22
23  try
24  {
25     Object[] resultList = (Object[])call.invoke(
26        new Object[]{"gov.nih.nci.cabio.domain.Gene", gene });
27     System.out.println("Total objects found: " + resultList.length);
28     if (resultList.length > 0)
29     {
30        for(int resultIndex = 0; resultIndex < resultList.length; resultIndex++)
31        {
32           Gene returnedGene = (Gene)resultList[resultIndex];
33           System.out.println(
34              "Symbol: " + returnedGene.getSymbol() + "\n" +
35              "\tName " + returnedGene.getFullName() + "\n" +
36              "");
37        }
38     }
39  } catch (Exception e) {
40     e.printStackTrace();
41  }
```

Lines	Description
1-2	Defines a new Web service call.
4-10	Maps a serialized object to its Java equivalent using the qualified name of the object from the WSDL file; in this case, the XML element Gene in the `urn:domain.cabio.nci.nih.gov` namespace is mapped to the Java Gene class.
12	Defines the service endpoint.
14-18	Sets the call properties including the name of the operation to invoke, the input parameters that will be sent and the return type to expect.
20-21	Creates a Gene criteria object and sets its symbol attribute to `"IL*"`; note that the `*.domain.*` package is used.

Lines	Description
25-26	Invokes the Web service operation using an array of two objects (target class name and criteria object) as input parameters and expecting an object array as its result.
32	Casts each object in the result array to type Gene.

Example Two: Searching Associations

This example is similar to the previous one but demonstrates how to search for associated elements by performing additional invocations of the query or queryObject operation.

```
1  try
2  {
3     Object[] resultList = (Object[])call.invoke(
4        new Object[]{"gov.nih.nci.cabio.domain.Gene", gene });
5     System.out.println("Total objects found: " + resultList.length);
6     if (resultList.length > 0)
7     {
8        for(int resultIndex = 0; resultIndex < resultList.length; resultIndex++)
9        {
10          Gene returnedGene = (Gene)resultList[resultIndex];
11          System.out.println(
12             "Symbol: " + returnedGene.getSymbol() + "\n" +
13             "\tName: " + returnedGene.getFullName() +
14             "");
15          Object[] nestedResultList = (Object[])call.invoke(
16             new Object[]{"gov.nih.nci.cabio.domain.Taxon", gene });
17          if (nestedResultList.length > 0)
18          {
19             Taxon returnedTaxon = (Taxon)nestedResultList[0];
20             System.out.println("\tTaxon: " + returnedTaxon.getScientificName());
21          }
22       }
23    }
24  } catch (Exception e) {
25     e.printStackTrace();
26  }
```

Lines	Description
15-16	A second operation invocation requests objects of type Taxon based on the same gene criteria used for the original query.
19	Casts the objects resulting from the nested query as Taxon objects.

Limitations

By default, the queryObject operation limits the result set to 1000 objects, even if the size of the result set is larger. To retrieve the objects past the 1000th record, you must use the query operation and specify the first object index (parameter in2) to be greater than 1000.

Result sets serialized and returned by the Web services interface do not currently include associations to related objects. A consequence of this behavior is that nested criteria objects with one-to-many associations that are passed to the query queryObject operations will result in an exception being thrown.

The following code demonstrates a Web services invocation that would fail:

```
1   Gene gene = new Gene();
2   gene.setSymbol("IL*");
3   Pathway pathway = new Pathway();
4   pathway.setId(new Long(120));
5   List pathwayList = new ArrayList()
6   pathwayList.add(pathway);
7   gene.setPathwaycollection(pathwayList);
8   try
8   {
9      Object[] resultList = (Object[])call.invoke
10         new Object[]{"gov.nih.nci.cabio.domain.Gene", gene })
11  } catch (Exception e)
12      // Web Services Exception will be caught
13  }
```

Because the Web services invocation has an inherent timeout behavior, queries that take a long time to execute may not complete. If this is the case, use the query method to specify a smaller result count.

XML-HTTP API

The caCORE XML-HTTP API, based on the REST (Representational State Transfer) architectural style, provides a simple interface using the HTTP protocol. In addition to its ability to be invoked from most internet browsers, developers can use this interface to build applications that do not require any programming overhead other than an HTTP client. This is particularly useful for developing web applications using AJAX (asynchronous JavaScript and XML).

Service Location and Syntax

The caBIO XML-HTTP interface uses the following URL syntax as described in *Table 4.3*.

```
http://{server}/{servlet}?query={returnClass}&{criteria}&
resultCounter={counter}&startIndex={index}&
pageSize={pageSize}&pageNumber={pageNumber}
```

Element	Meaning	Required	Example
server	Name of the web server on which the caBIO web application is deployed.	Yes	cabio.nci.nih.gov
servlet	URI and the name of the servlet that will accept the HTTP GET requests	Yes	cabio43/server/ GetXML cabio43/server/ GetHTML
returnClass	Class name indicating the type of objects that this query should return	Yes	query=Gene
criteria	Search request criteria describing the requested objects	Yes	Gene[@id=2]

Table 4.3 URL syntax used by the caBIO XML-HTTP interface

Element	Meaning	Required	Example
counter	Number of top level objects returned by the search	No	resultCounter=500
index	Start index of the result set	No	startIndex=25
pageSize	Number of records to display on each "page	No	pageSize=50
pageNumber	The number of the "page" for which to display results	No	pageNumber=3

Table 4.3 URL syntax used by the caBIO XML-HTTP interface

The caBIO architecture currently provides two servlets that accept incoming requests:

- **GetXML** returns results in an XML format that can be parsed and consumed by most programming languages and many document authoring and management tools

- **GetHTML** presents result using a simple HTML interface that can be viewed by most modern Internet browsers

Within the request string of the URL, the criteria element specifies the search criteria using XQuery-like syntax (*Table 4.4*). Within this syntax, square brackets ("[" and "]") represent attributes and associated roles of a class, the "at" symbol ("@") signals an attribute name/value pair, and a forward slash character ("/") specifies nested criteria. Criteria statements within XML-HTTP queries are generally of the following forms (although more complex statements can also be formed):

```
{ClassName}[@{attributeName}={value}] [@{attributeName}={value}]…
```

```
{ClassName}[@{attributeName}={value}]/
    {ClassName}[@{attributeName}={value}]/…
```

Parameter	Meaning	Example
ClassName	The name of a class	Gene
attributeName	The name of an attribute of the return class or an associated class	symbol
Value	The value of an attribute	brca*

Table 4.4 Criteria statements within XML-HTTP queries

Examples of Use

The following examples demonstrate use of the XML-HTTP interface. In actual use, the queries shown here would either be submitted by a block of code or entered in the address bar of an Internet browser. Also note that the servlet name GetXML in each of the examples can be replaced with GetHTML to view with layout and markup in a browser.

Query	http://server/servlet/GetXML?query=Gene&Gene[@symbol=brca*]
Syntactic Meaning	Find all objects of type `Gene` whose symbol starts with 'brca'.
Biological Meaning	Find all BRCA genes.

Query	http://server/servlet/GetXML?query=Gene&Gene[@symbol=brca*]/Taxon[@scientificName=homo sapiens]
Syntactic Meaning	Find all objects of type `Gene` whose symbol starts with 'brca' and which have an associated Taxon object whose scientificName is equal to 'homo sapiens'.
Biological Meaning	Find all human BRCA genes.

Query	http://server/servlet/GetXML? query=Tissue&Tissue[@organ=eye][@histology=neoplasia]
Syntactic Meaning	Find all objects of type `Tissue` associated with attribute organ equal to 'eye' and histology equal to 'neoplasia'.
Biological Meaning	Find all tissues representing neoplasms of the eye.

Query	http://server/servlet/GetXML? query=Gene&Chromosome[@number=2]/Taxon[@scientificName=homo sapiens]
Syntactic Meaning	Find all objects of type `Gene` associated with `Chromosome` objects with number equal to 2 which themselves are related to `Taxon` objects with `scientificName` equal to 'homo sapiens' in biological terms
Biological Meaning	Find all human genes on chromosome number 2.

Working With Result Sets

Because HTTP is a stateless protocol, the caBIO server has no knowledge of the context of any incoming request. Consequently, each invocation of `GetXML` or `GetHTML` must contain all of the information necessary to retrieve the request, regardless of previous requests. Developers should consider this when working with the XML-HTTP interface.

Retrieving Related Results using XLinks

When using the GetXML servlet to retrieve results as XML, associations between objects are converted to XLinks within the XML. The link notation, shown below, allows the client to make a subsequent request to retrieve the associated objects.

```
<class name="gov.nih.nci.cabio.domain.Gene" recordNumber="1">
     …
     <field name="taxon"
        xlink:type="simple"
     xlink:href=

"http://cabioapi.nci.nih.gov/cabio43/GetXML?query=Taxon&Gene[@id=5]">

            getTaxon
        </field>
   …
</class>
```

Controlling the Number of Items Returned

The GetXML servlet provides a throttling mechanism to allow developers to define the number of results returned on any single request and where in the result set to start. For example, if a search request yields 500 results, specifying `resultCounter=450` returns only the last 50 records. Similarly, specifying `startIndex=50` returns only the first 50 records.

Paging Results

In addition to controlling the number of results to display, the GetXML servlet also provides a mechanism to support "paging". This concept, common to many web sites, allows results to be displayed over a number of pages, so that, for example, a request that yields 500 objects could be displayed over 10 pages of 50 objects each. When the paging feature is used, the GetXML servlet will include XLinks to each of the result pages in an XML <page/> element. The element data of the <page/> element is the number of the page, suitable for output as text or HTML when using an XSL stylesheet:

```
  <page number="1"
       xlink:type="simple"
       xlink:href="http://cabioapi.nci.nih.gov/cabio43/GetXML?query=
{query}&pageNumber=4&resultCounter=1000&startIndex=0"> 4 </page>
```

Limitations

When specifying attribute values in the query string, use of the following characters generates an error: [ ] / \ # & %.

Grid Service API

The caBIO Grid Service API provides access to caBIO data through the caBIG® grid (caGrid).

Service Location

caBIO Grid Services can be accessed via the following URL:

V 4.0: http://cabiogrid40.nci.nih.gov:80/wsrf/services/cagrid/CaBIO40GridSvc

Note: caBIO 3.1 and 3.2 were retired at the end of June 2009.

These grid services can also be accessed through the caGrid Portal.

Examples of Use

caBIO grid services can be invoked using the caGrid Common Query Language (CQL). Below are examples of CQL queries for caBIO data.

Example One: Return all Genes

```
<CQLQuery xmlns="http://CQL.caBIG/1/gov.nih.nci.cagrid.CQLQuery">
   <Target name="gov.nih.nci.cabio.domain.Gene">
   </Target>
</CQLQuery>
```

Example Two: Return all Genes with a Symbol Beginning with "BRCA"

```
<CQLQuery xmlns="http://CQL.caBIG/1/gov.nih.nci.cagrid.CQLQuery">
   <Target name="gov.nih.nci.cabio.domain.Gene">
      <Attribute name="symbol" predicate="LIKE" value="BRCA%"/>
   </Target>
</CQLQuery>
```

Example Three: Return all Genes with a Symbol Beginning with "BRCA" and an Associated Taxon with a Scientific Name Equal to "Homo sapiens"

```
<CQLQuery xmlns="http://CQL.caBIG/1/gov.nih.nci.cagrid.CQLQuery">
   <Target name="gov.nih.nci.cabio.domain.Gene">
     <Group logicRelation="AND">
        <Attribute name="symbol" predicate="LIKE" value="BRCA%"/>
           <Association roleName="taxon"
name="gov.nih.nci.cabio.domain.Taxon">
                 <Attribute name="scientificName" predicate="EQUAL_TO"
value="Homo sapiens"/>
           </Association>
     </Group>
    </Target>
</CQLQuery>
```

For examples on how to invoke caGrid Data Services via the Java programming language, refer to Chapter 6 of the *caGrid 1.1 Programmer's Guide*. Detailed instructions and examples for discovering and accessing caBIO grid services, are also available on the caGrid Wiki: http://www.cagrid.org/mwiki/index.php?title=CaGrid.

Working With Result Sets

Results from CQL queries are returned as XML. Below is an example of the CQL query results from a caBIO gene query:

```xml
<ns1:CQLQueryResults targetClassname="gov.nih.nci.cabio.domain.Gene"
   xmlns:ns1="http://CQL.caBIG/1/gov.nih.nci.cagrid.CQLResultSet">
- <ns1:ObjectResult>
   <ns2:Gene bigid="hdl://2500.1.PMEUQUCCL5/HEGOZJZIY3" clusterId="2"
    fullName="N-acetyltransferase 2 (arylamine N-acetyltransferase)"
    symbol="NAT2" id="2"
    xmlns:ns2="gme://caCORE.caBIO/4.0/gov.nih.nci.cabio.domain" />
   </ns1:ObjectResult>
- <ns1:ObjectResult>
   <ns3:Gene bigid="hdl://2500.1.PMEUQUCCL5/S344L2X73O" clusterId="4"
    fullName="Alcohol dehydrogenase 1B (class I), beta polypeptide"
    symbol="ADH1B" id="3"
    xmlns:ns3="gme://caCORE.caBIO/4.0/gov.nih.nci.cabio.domain" />
   </ns1:ObjectResult>
- <ns1:ObjectResult>
   <ns4:Gene bigid="hdl://2500.1.PMEUQUCCL5/I62AZYNECK"
    clusterId="11" fullName="Carcinoembryonic antigen-related cell
    adhesion molecule 3" symbol="CEACAM3" id="4"
    xmlns:ns4="gme://caCORE.caBIO/4.0/gov.nih.nci.cabio.domain" />
   </ns1:ObjectResult>

...

- <ns1:ObjectResult>
   <ns1001:Gene bigid="hdl://2500.1.PMEUQUCCL5/TWBNWHRUV4"
    clusterId="30213" fullName="Ceroid-lipofuscinosis, neuronal 5"
    symbol="CLN5" id="1544"
    xmlns:ns1001="gme://caCORE.caBIO/4.0/gov.nih.nci.cabio.domain"
    />
   </ns1:ObjectResult>
   </ns1:CQLQueryResults>
```

Limitations

caBIO Grid services for a release are deployed after the caBIO API release has been deployed to Production to allow for time required to generate, test, and deploy new grid services for the release. To obtain the latest version of caBIO grid services, leverage the [caBIO Grid Portal](#) to discover caBIO services.

caBIO FreeStyleLM and Range Query search facilities are currently not available as grid services.

CANCER BIOINFORMATICS INFRASTRUCTURE OBJECTS

This chapter describes the Cancer Bioinformatics Infrastructure Objects (caBIO) model and its application programming interfaces.

Topics in this chapter include:

- *Introduction* on this page

- *caBIO API* on page 58

- *Data Sources in the caBIO Database* on page 71

- *caGrid Identifiers* on page 78

- *Classes Indexed for FreeStyleLM Search* on page 78

Introduction

The Cancer Bioinformatics Infrastructure Objects (caBIO) model and architecture was the first of several model-driven information systems that make up caCORE and continues to be an on-going effort to model the genomic domain. The caBIO objects simulate the behavior of actual genomic components such as genes, chromosomes, sequences, libraries, clones, ontologies, etc. They provide access to a variety of genomic data sources including GenBank, Unigene, LocusLink, Homologene, Ensemble, UCSC Genome Sequencing Center's Annotations, Ensembl's Compara, Canada Drugbank, NCI's CTEP (Cancer Therapy Evaluation Program) and NCI CBIIT's CGAP (Cancer Genome Anatomy Project) data repositories. The full list of data sources is listed starting on page 71.

caBIO API

Most of the domain objects defined in the caBIO API are objects that specialize in bioinformatics applications. The caBIO domain objects are implemented as Java beans in the `gov.nih.nci.cabio.domain`, `gov.nih.nci.common.domain`, `gov.nih.nci.cabio.pathway`, and `gov.nih.nci.common.provenance.domain` packages and include those classes that correspond to biological entities and bioinformatics concepts.

The caBIO UML model is published as an EA (Enterprise Architect) in the Files section of the caBIO GForge site (https://gforge.nci.nih.gov/frs/?group_id=51). *Table 5.1* and *Table 5.2* list and describe the classes in these packages. Detailed descriptions about each class and its methods are present in the caBIO 4.3 JavaDocs (http://cabioapi.nci.nih.gov/cabio43/docs/. .

caBIO Domain Object	*Description*
Agent	An object representing a therapeutic agent (drug, intervention therapy). It provides cross references, pharmacology and pharmacokinetic annotations.
	Data in the Agent object includes static agents associated with clinical trials, agents associated with genes based on the Cancer Gene Index, and drugs from the Canada Drugbank.
	Depending on the data source the agent comes from, it may be associated with clinical trials (Clinical TrialProtocol) and genes (Target, GeneAgentAssociation). It may also have aliases (AgentAlias) and associated genes (GeneAgentAssociation).
AgentAlias	Represents an alias for a therapeutic agent. For example, drug synonyms and trade names can be found here.
Anomaly	An irregularity in either the expression of a gene or its structure (i.e., a mutation). It provides access to the associated histopathological information (`Histology`) as well as the disease (`DiseaseOntology`) and organ ontology (`OrganOntology`).
	A user can also retrieve the anomalies (`Anomaly`) associated with a given gene target using the `getAnomalyCollection` functionality.
	The data in Anomaly is currently static.

Table 5.1 caBIO domain objects and descriptions

caBIO Domain Object	Description
ArrayReporter	A superclass for different kinds of microaarray reporters such as `ExonArrayReporter`, `SNPArrayReporter`, `TranscriptArrayReporter` and `ExpressionArrrayReporter`. These objects get populated with annotations from Affymetrix, Illumina and Agilent Arrays such as the Affymetrix HG-U133, HG-U133 Plus2, HT-HG-U133 A, HT-HG-U133 A2.0 Arrays, HuMapping 250kNsp, HuMapping 250kSty, HuMapping 50kHind240, HuMapping Xba240 and Exon Arrays, Agilent 44K and Acgh 244k arrays and Illumina HumanHap Arrays and are synchronized monthly with the latest releases of the respective datasets. In case of SNP Arrays such as Affymetrix HuMapping and Illumina arrays, the location of a SNP relative to a Marker is available through `MarkerRelativeLocation` and relative through a Gene is available through `GeneRelativeLocation`.
ArrayReporterCytogenetic Location	Provides cytogenetic starts and stops for array reporters in ArrayReporter, where available. It exposes cytogenetic starts and stops for Affymetrix HG-U133, HG-U133 Plus2, HT-HG-U133A, HG-HG-U133A 2.0, HuMapping 250k Nsp, HuMapping 250k Sty, HuMapping 50k Hind240, HuMapping 50k Xba 240, Agilent 44K and Agilent CGH 244K arrays and gets populated with data provided by the manufacturer.
ArrayReporterPhysical Location	Provides chromosomal starts and stops for array reporters in `ArrayReporter`. It exposes chromosomal starts and stops for Affymetrix HG-U133, HG-U133 Plus2, HT-HG-U133A, HG-HG-U133A 2.0, HuMapping 250k Nsp, HuMapping 250k Sty, HuMapping 50k Hind240, HuMapping 50k Xba 240, Exon Arrays, Agilent 44K and Agilent aCGH 244K arrays and Illumina Human Hap Arrays and gets populated with data provided by the manufacturer.
BiochemicalReaction	A type of pathway interaction representing an interaction among physical entities, and treated as a single event. It is populated with data from PID.
Chromosome	An object representing a specific chromosome in an organism; It provides access to all known genes on the chromosome, as well as to the `ChromosomalLocation` and `CytogeneticLocation` of features such as SNP, ESTs, mRNAs, Markers, ArrayReporters, etc through the `getLocationCollection` functionality. This ability to associate multiple features with a chromosome aids in performing range-query searches, described in greater detail elsewhere. The data in the chromosome table is static and as with data in caBIO, encompasses only human and mouse chromosomes.

Table 5.1 caBIO domain objects and descriptions (Continued)

caBIO Domain Object	Description
ClinicalTrialProtocol	The protocols and administrative information about the trial such as lead organization, participants, current phase of the clinical trial, and associated NIH Administrative Id.
	A `ClinicalTrialProtocol` object provides the user access to the associated agents, histopathology and disease ontologies through the `getAgentCollection`, `getDiseaseOntologyCollection` and `getHistopathologyCollection` functionalities respectively. This object gets populated with data from NCI's CTEP Program which gets updated on a monthly basis.
Clone	An object used to hold information pertaining to I.M.A.G.E. clones associated with human and mouse genes from Unigene's repository. A user can access the associated sequences (`NucleicAcidSequence`) using the `getNucleicAcidSequenceCollection` functionality.
	A clone is always associated with a clone-library further details of which are available through the associated Library Object.
	The orientation of the clone along with the associated sequence identifier is available through the `CloneRelativeLocation` object that provides further information regarding the clone insert read.
	The Clone object gets populated with clone data from Unigene's human and mouse annotations and that can be associated to existing libraries in caBIO's Library Object.
	As indicated below, the Library object in turn gets populated with clone-library data from CGAP who integrate information from ORESTES, dbEST, Unigene and other such resources.
CloneRelativeLocation	Provides the orientation (5'/3') of a clone insert read, along with links to the associated sequence information (`NucleicAcidSequence`).
	It gets populated with relative location of clones in the Clone Object as available in Unigene.
ComplexComponent	A pathway participant (subclass of `PhysicalParticipant`) which is part of a complex (`ComplexEntity`). It is populated with data from PID.
ComplexEntity	A pathway participant (subclass of `PhysicalParticipant`) which is part of a complex (`ComplexEntity`). It is populated with data from PID.
	A type of pathway entity (subclass of `PhysicalEntity`) which is composed of pathway participants (`ComplexComponents`). It is populated with data from PID.

Table 5.1 caBIO domain objects and descriptions (Continued)

caBIO Domain Object	Description
Condition	A pathway participant (subclass of `PhysicalParticipant`) which is part of a complex (`ComplexEntity`). It is populated with data from PID.
	A type of pathway entity (subclass of `PhysicalEntity`) which is composed of pathway participants (`ComplexComponents`). It is populated with data from PID.
	A kind of pathway participant (subclass of `Participant`) representing a prerequisite condition for an interaction to occur. It is populated with data from PID.
ConstrainedRegion	A highly conserved region on the human genome as indicated by an n-way multiple alignment analysis from Ensembl Compara.
Cytoband	The `Cytoband` Object provides information such as the name and location of cytobands in human and mouse available from UCSC's Genome Sequencing Center for every new build of the Human and Mouse Genome.
	It also gets populated with human cytoband information available in Unigene, Affymetrix and Agilent array probeset to facilitate retrieval of cytobands that identify the start and stop of a SNP, Gene and ArrayReporter in `SNPCytogeneticLocation`, `GeneCytogeneticLocation` and `ArrayReporterCytogeneticLocation` respectively.
CytobandPhysicalLocation	An abstract class that provides the chromosomal starts and stops of Cytobands. It gets populated with data from UCSC's Genome Sequence Center.
CytogeneticLocation	An abstract superclass that provides cytogenetic starts and stops for different features such as SNP, Gene and ArrayReporters through their respective subclasses, `SNPCytogeneticLocation`, `GeneCytogeneticLocation` and `ArrayReporterCytogeneticLocation` respectively.
	Each of these subclasses gets populated with data from different sources as indicated below.
DatabaseCrossReference	An object that cross-references caBIO `Genes`, `SNPs`, and `Proteins` with identifiers from different databanks.
	For example, it associates caBIO genes (Unigene) to Entrez and OMIM, caBIO SNPs (dbSNP) to TSC and caBIO Proteins (Uniprot) to corresponding identifiers in Ensembl and Enzyme Commission.
	It uses references to Entrez-Gene and LocusLink in Unigene's human and mouse data to associate Unigene, and uses Entrez-based datasets to associate caBIO Genes with OMIM.
	It uses references to Ensembl and Enzyme Commission Ids from Affymetrix SNP array-annotations to associate caBIO Proteins with Ensembl, RefSeq and EC identifiers.

Table 5.1 caBIO domain objects and descriptions (Continued)

caBIO Domain Object	Description
DiseaseOntology	An object representing various oncological disease ontologies. It provides access to the clinical trials associated with these diseases (`ClinicalTrialProtocol`) and the associated histopathological (`Histopathology`). Gene-Disease associations made through the Cancer Gene Index Project are available through the `getGeneFunctionAssociation` functionality. Apart from existing static data, this object gets populated with data from CGDC Data.
DiseaseOntology Relationship	Provides a hierarchical relationship among various diseases and associated ontologies. Thus it allows a user to obtain `DiseaseOntologys` that share a parent/child relationship. The data in this table is static.
EntityAccession	The accession number or native database id of an entity in a pathway. It is populated with data from PID.
EntityName	The name of an entity in a pathway. It is populated with data from PID.
Evidence	Provides details of the scientific publication used to make the Gene-Agent or Gene-Disease association through text-mining, as part of NCI's CGDC Project. It provides details such as the Pubmed identifier, the sentence(s) from the publication highlighting the association, the status of the curation process, the NCI-provided Evidence Code assigned to this Evidence (that summarizes the role of the Gene in the disease or the interaction of the Agent with the Gene). It provides access to further details about the assigned EvidenceCode through the `getEvidenceCode` functionality.
EvidenceCode	Provides access to NCI-curated Evidence Codes to represent status of Gene-Disease and Gene-Agent associations made through the CGDC Project. It gets populated with data from CGDC and provides a link to the associated Evidence (PubMed publications used to form the Gene-Agent or Gene-Disease associations through text-mining)
Exon	An `Exon` represents an expressed part of a gene. In caBIO, an exon constitutes part of a `Transcript` that is used as a probe in Affymetrix Exon Arrays. It gets populated with data from Affymetrix Human Exon Arrays. Transcripts associated with this Exon, per Affymetrix Exon Array annotations are available through the `Transcript` object, while the probesets themselves are available through ArrayReporter's `ExonArrayReporter` object.

Table 5.1 caBIO domain objects and descriptions (Continued)

caBIO Domain Object	Description
ExonArrayReporter	One of the two 'transcript-array' objects in caBIO, the other being `ExpressionArrayReporter`. An object representing Affymetrix Exon Arrays providing details such as the manufacturer's PSR Id, the probe count and the strand information for each probeset. A user can retrieve the associated Exons, Transcripts and Genes using the `getTranscriptCollection` and `getGeneCollection` methods respectively.
ExpressedSequenceTag	A subclass of `NucleicAcidSequence`. It represents short nucleotide sequences (typically 400-600 bases) from one or both ends of randomly selected cDNA clones and gets populated with data from Unigene's human and mouse datasets. It provides access to the same functionalities such as available through the parent `NucleicAcidSequence` class, such as clones, relative locations of these Clones, the chromosomal start and stop (`NucleicAcidPhysicalLocation`), corresponding Genes and hence any associated `ExpressionArray` reporters.
ExpressionArrayReporter	Gene objects representing mRNA-based arrays in caBIO. It provides information on probesets in the Affymetrix Human U133, HG U133 Plus2, HT HG U133 A and HT HG U133 A2.0 Arrays, and `NucleicAcidSequence` respectively. A user can retrieve the associated Genes and protein domains (`ProteinDomain`) through the `getGeneCollection` and `getProteinDomainCollection` methods respectively. The sequence information for the representative Mrna sequence of this array-probe can be obtained from NucleicAcidSequence using `getNucleicAcidSequenceCollection` method. The mRNA-transcripts used to form the array probe itself is encapsulated in the Transcript object, along with such transcripts from Affymetrix Human Exon Arrays.

Table 5.1 caBIO domain objects and descriptions (Continued)

caBIO Domain Object	*Description*
Gene	Gene objects are the effective portal to most of the genomic information provided by the caBIO data services; It provides access to the following objects: a. associated proteins (`Protein`) b. gene-targets (`Target`) c. organs for which histopathological information of tumor-samples is available that implicates the gene (`OrganOntology`) d. histopathologies (`Histopathology`) associated with c) e. curated gene-ontologies (`GeneOntology`) f. BioCarta pathways (`Pathway`) involving the gene g. aliases (`GeneAlias`) from HUGO and Entrez h. clones (`Clone`) from Unigene i. Mrna/EST-sequences (`NucleicAcidSequence`) j. SNPs located on the Gene (`GeneRelativeLocation`) as per Affymetrix array annotation data files. k. chromosomal start and stop (`GenePhysicalLocation`) from MapView l. Agents it interacts with, per NCI's CGDC Text Mining Project (`GeneAgentAssociation`) m. Tumors it is implicated in (`GeneDiseaseAssociation`) as per NCI's CGDC Text Mining Project. n. Corresponding Identifiers from OMIM and Entrez in `DatabaseCrossReference` o. UniSTS Markers corresponding to this gene in the Marker object p. Symbols from HUGO The Gene object itself is populated with human and mouse data from Unigene. Other associated objects (and hence their associations to Gene) get populated from different sources.
GeneAlias	An alternative name for a gene; provides descriptive information about the gene (as it is known by this alias), as well as access to the primary Gene it refers to. It gets populated with data from HUGO, Entrez and CGAP
GeneAgentAssociation	Agents interacting with a Gene as per results of text mining scientific publications from PubMed in the CGDC Project, or genetic target associations from the Canada DrugBank. It provides an association to details of the corresponding publications through the Evidence object. It is a subclass of the `GeneFunctionAssociation` class.
GeneCytogeneticLocation	Start and stop of the Gene in terms of Cytobands. It provides details of the cytoband through the associated Cytoband object. It gets populated with data from Unigene.

Table 5.1 caBIO domain objects and descriptions (Continued)

caBIO Domain Object	Description
GeneDiseaseAssociation	DiseaseOntologies associated with a Gene as per results of text mining scientific publications from PubMed in the CGDC Project. It provides an association to details of the corresponding publication through the `Evidence` object. It is a subclass of the `GeneFunctionAssociation` class.
GeneFunctionAssociation	An abstract superclass of the `GeneDiseaseAssociation` and `GeneAgentAssociation` classes.
GenePhysicalLocation	An object representing chromosomal starts and stops of genes. It gets populated with data from NCBI's MapView application
GeneOntology	An object providing information on a Gene's position in the Gene Ontology Consortium's controlled vocabularies. It provides access to Genes corresponding to an ontological term and to the relevant parent/child ontological relationship, if any via the corresponding `GeneOntologyRelationship`. It gets populated with GO data from CGAP and is kept synchronized with the parent GO Consortium datasets.
GeneOntology Relationship	Specifies the parent/child relationship, if any, for a given `Gene Ontology` and allows the user to retrieve such associated `GeneOntologys`. It gets populated with data from CGAP, as with the `GeneOntology` object above.
GeneRegulation	A type of pathway interaction (subclass of Interaction) that regulates gene expression. It is populated with data from PID.
GeneRelativeLocation	An object providing location of SNPs on Genes such as whether the SNP is located in the 3' UTR, 5' UTR, coding region of the `Gene`. It gets populated with data from Affymetrix Human Mapping 250K Nsp and Sty, 50K Xba240 and Hind240 and Illumina 550K genotyping arrays and provides a link to the associated Gene.
Histopathology	Represents anatomical changes in a diseased tissue sample associated with an expression experiment; captures the relationship between an organ and disease. A user can also retrieve any associated Pathways.
Homologous Association	Represents degrees of homology (as a percentage) amongst various genes in human and mouse, along with access to further details about the Genes themselves. It is populated with human and mouse data from CGAP.
Input	A pathway participant (subclass of `PhysicalParticipant`) that is consumed by an interaction. It is populated with data from PID.
Interaction	A biological event which is part of a larger process (`Pathway`). It is populated with data from PID.

Table 5.1 caBIO domain objects and descriptions (Continued)

caBIO Domain Object	*Description*
Library	An object representing a CGAP library; provides access to information about: the tissue sample and its method of preparation, the library protocol that was used, the clones contained in the library, and the sequence information derived from the library.
	CGAP in turn amalgamates data from a variety of different Clone Libraries such as dbEST, ORESTES, Unigene, etc.
Location	Super class for `<Feature>PhysicalLocation` and `<Feature>CytogenicLocation` objects. Provides start and stop locations interms of chromosomes and cytobands for
	a. Genes : `GenePhysicalLocation` and `GeneCytogeneticLocation`
	b. SNPs : `SNPPhysicalLocation` and `SNPCytogeneticLocation`
	c. Markers : `MarkerPhysicalLocation`
	d. Exons : `ExonPhysicalLocation`
	e. Transcripts : `TranscriptPhysicalLocation`
	f. ArrayReporters : `ArrayReporterPhysicalLocation` and `ArrayReporterCytogeneticLocation`
	g. `NucleicAcidSequnce` (ESTs and MRNAs) : `NucleicAcidPhysicalLocation`
	h. Cytobands : `CytobandPhysicalLocation`
Macroprocess	A type of pathway interaction (subclass of `Interaction`) annotated with controlled vocabulary terms from the Gene Ontology's Biological Process vocabulary. It is populated with data from PID.
Marker	An object representing different kinds of Markers such as 5'/3' Primers, Microsatellites and SNPs.
	It gets populated with human and mouse primers from UniSTS, microsatellite markers from Affymetrix HuMapping Arrays and SNPs from TSC.
	The location of a SNP relative to a marker is available through `MarkerRelativeLocation` and is populated with data from Affymetrix HuMapping arrays.
	Chromosomal start and stop of a Marker is available through `MarkerPhysicalLocation`
MarkerAlias	An object representing different aliases or names for a Marker. It gets populated with data from UniSTS
MarkerPhysicalLocation	Class that provides the chromosomal start and stop of the Marker. It gets populated with data from NCBI's MapView.
MarkerRelativeLocation	An object providing location of SNP from Affymetrix HuMapping Arrays relative to a Microsatellite Marker from these same arrays.
	The associated probeset information is available through `ArrayReporter`

Table 5.1 caBIO domain objects and descriptions (Continued)

caBIO Domain Object	Description
Microarray	An object representing various micrrrays in caBIO such as the Affymetrix Human Mapping, HG-U133, HG-U133 Plus2, HT-HG-U133, HT-HG-U133 A 2.0 and Illumina HumanHap 550K 'SNP'/genotyping arrays, Affymetrix 'Exon' Arrays and 'oligo' arrays such as the Agilent Human Genome 44k and aCGH 244k arrays.
MultipleAlignment	An evolutionary analysis of an n-way multiple alignment, from Ensembl Compara. This object is associated with many ConstrainedRegion objects which are the output of the analysis, and a set of Taxon objects indicating the species which were used in the alignment.
NegativeControl	A type of pathway participant (subclass of `PhysicalParticipant`) that exercises negative control over an interaction, but is not consumed by the interaction. It is populated with data from PID.
NucleicAcidPhysical Location	Provides chromosomal start and stop of an mRNA or an EST, as provided by UCSC's Genome Sequencing Center.
NucleicAcidSequence	An object representation of a gene sequence; provides access to the clones from which it was derived, the ASCII representation of the residues it contains, and any `ExpressionArrayReporters` for which it is a representative sequence. It is populated with data pertaining to human and mouse from Unigene. Its chromosomal start and stop is available through `NucleicAcidPhysicalLocation`
OrganOntology	A representation of an organ whose name occurs in a controlled vocabulary; provides access to any associated histopathological information, and anomalies. Because it is "ontolog-able", It provides access to any associated ontologies that it shares a parent/child relationship with, through the linked `OrganOntologyRelationship`. Data in this table is static.
OrganOntologyRelationship	An object that describes parent/child relationships, if any, amongst `OrganOntologys`. Data in this table is static.
Output	A type of pathway participant (subclass of `PhysicalParticipant`) which is produced by an interaction. It is populated with data from PID.
Participant	A participant in a pathway interaction or part of a complex. May be a Condition or a PhysicalEntity. It is populated with data from PID.

Table 5.1 caBIO domain objects and descriptions (Continued)

caBIO Domain Object	Description
Pathway	An object representation of a molecular/cellular pathway in human and mouse compiled by BioCarta. It provides access to further details about the associated Genes and any available histopathological information for genes involved in clinical trials through `getGeneCollection` and `getHistopathologyCollection` methods. It gets populated with data from CGAP and is composed of BioCarta Pathways. It provides links to genes associated with every pathway as well as histopathological information pertaining to these Genes through the associated Histopathology object.
PathwayReference	A kind of pathway interaction (subclass of `Interaction`) which stands for another pathway. It is populated with data from PID.
PhysicalEntity	A reference to a molecular species (protein, complex, RNA, or small molecule) that may participate in a pathway. It is populated with data from PID.
PhysicalEntityFamily	A set of related molecular species (`PhysicalEntity` objects) constituting a family. It is populated with data from PID.
PhysicalLocation	Provides chromosomal start and stop positions for SNPs, ESTs, MRNAs, Markers and Cytobands. Further details for the SNP can be obtained from the SNP object. The sequence associated with the ESTs and MRNAs can be obtained from the linked `NucleicAcidSequence` object. The SNP Data is obtained from NCBI's dbSNP, while the latter are obtained from UCSC's Genome Sequencing Center for every Human and Mouse Genome build.
PhysicalParticipant	A condition or physical entity that participates in an interaction or is a component of a complex (`ComplexEntity`). It is populated with data from PID.
PopulationFrequency	Represents the major and minor alleles of a SNP and their respective frequencies in different populations.
PositiveControl	A pathway participant (subclass of `PhysicalParticipant`) that exercises a positive (promoting) control over an interaction but is not consumed by the interaction. It is populated with data from PID.
Protein	An object representation of a protein; provides access to the encoding Gene via its GenBank ID, the Taxon in which this instance of the protein occurs, and references to homologous proteins in other species. It is populated with data from Uniprot-Swissprot.
ProteinAlias	An alternate name for a protein. This data is obtained from Uniprot-Swissprot.

Table 5.1 caBIO domain objects and descriptions (Continued)

caBIO Domain Object	*Description*
ProteinDomain	An object representing protein domains from Interpro. This object is populated with Interpro protein domains associated with probesets from Affymetrix HG-U133, HG-U133 Plus2, HT-HG-U133 and HT-HG-U133A 2.0 Array annotations.
ProteinEntity	A kind of physical entity (subclass of `PhysicalEntity`), consisting of a sequence of amino acids. This object serves as a reference to one or more Proteins, for use within a Pathway. It is populated with data from PID.
ProteinSequence	The sequence of a protein from Swissprot.
ProteinSubunit	A cleaved product of a protein, for use within a `Pathway`. It is populated with data from PID.
Protocol	An object representation of the protocol used in assembling a clone library. It is populated with data from NCI's CTEP (Cancer Therapy Evaluation Program)
ProtocolAssociation	An association class relating `ClinicalTrialProtocols` to Diseases.
RelativeLocation	An abstract super class for `MarkerRelativeLocation` and `GeneRelativeLocation` that provides the relative locations of SNPs with regards to Genes and Markers. These classes get populated with data from Affymetrix HuMapping arrays and Illumina HapMap SNP Arrays.
RNAEntity	A kind of physical entity (subclass of `PhysicalEntity`), which is an RNA macromolecule. This object serves as a reference to one or more `NucleicAcidSequence` objects, for use within a `Pathway`. It is populated with data from PID.
SmallMoleculeEntity	A kind of physical entity (subclass of `PhysicalEntity`), which is not a macromolecule (i.e. coded by DNA) and is not a complex. This object serves as a reference to one or more `Agent` objects, for use within a `Pathway`. It is populated with data from PID.
SNP	An object representing a Single Nucleotide Polymorphism; It provides the two most common substitutions at that position and the offset of the SNP in the parent sequence. A reference to related SNPs from The SNP Consortium (TSC) is available through `DatabaseCrossReference`. It provides a link to the associated Population Frequencies. The location of a SNP is available through the `PhysicalLocation` and `Location` objects as well as through the `SNPPhysicalLocation` object. Location of SNPs in terms of cytogenetic starts and stops is available through the `SNPCytogeneticLocation` object.
SNPArrayReporter	An object representing probeset information for genotyping arrays such as Affymetrix Human Mapping Arrays and Illumina HumanHap 550k Arrays; It provides details on the corresponding SNP through the linked SNP object.

Table 5.1 caBIO domain objects and descriptions (Continued)

caBIO Domain Object	Description
SNPCytogeneticLocation	Provides start and stops for SNPs in the SNP object in terms of cytobands. It gets populated with data from Affymetrix HuMapping array annotations.
SNPPhysicalLocation	Provides chromosomal start and stop for SNPs in the SNP object. Gets populated with data from NCBI's dbSNP
Target	A gene thought to be at the root of a disease etiology, and which is targeted for therapeutic intervention in a clinical trial.
	Data in this object is static. Each target links to the associated Gene object through which further details of the Target are available.
Taxon	An object representing the various names (scientific, common, abbreviated, etc.) for human and mouse genomes. It can be used to retrieve the associated Genes, Chromosomes, Pathways if any, Proteins, or Tissues.
	This object contains a static set of taxons, as well as additional data loaded from Ensembl Compara as necessary to support the MultipleAlignment-Taxon association.
Tissue	A group of similar cells united to perform a specific function.
	Data in this object is static
Transcript	An object representation of a collection of Exons, that constitutes a probe in an Affymetrix Exon Array. It can be used to retrieve the associated `Exons`, `ExonArrayReporters` and `Genes`. It gets populated with data from Affymetrix HuExon Arrays
TranscriptArrayReporter	An abstract superclass for `ExpressionArrayReporter` and `ExonReporter`.
TranscriptPhysicalLocation	Chromosomal Start and Stop of a Transcript obtained from Affymetrix HuExon Arrays
Vocabulary	A collection of terms associated with a given Anomaly.
	Data in this object is static

Table 5.1 caBIO domain objects and descriptions (Continued)

Provenance Objects	Description
Provenance	An object modeling the provenance of various entities such as SNP, Gene, NucleicAcidSequence and Protein data in caBIO. It exposes information such as the source of the data (dbSNP, Unigene, LocusLink, Uniprot, etc) along with the class name modeling that object in caBIO, evidence code, etc.
InternetSource	An object modeling details of the primary data source providers for SNP, Gene, etc in caBIO
PublicationSource	An object modeling details of publications associated with caBIO data such as the author, date of publication, etc.

Table 5.2 Provenance objects and descriptions

Provenance Objects	Description
ResearchInstitutionSource	An object modeling details of the research institution such as name, address, contact information, etc from where caBIO data is obtained.
Source	An object modeling the primary data source for objects like SNP, Gene, Protein, etc in caBIO
SourceReference	An object providing web-enabled access to further details regarding Genes, Proteins, SNPs, and other caBIO objects from their primary data sources.
URLSourceReference	An object providing web-enabled access to obtain further details regarding Genes, Proteins, SNPs, and other caBIO objects from their primary data sources.

Table 5.2 Provenance objects and descriptions (Continued)

Data Sources in the caBIO Database

This section describes the internal and external data sources for caBIO and how the information these sources provide can be accessed via caBIO objects.

The caBIO application programming interfaces were developed primarily in response to the need for programmatic access to the information at several NCI web sites, including:

- Cancer Genome Anatomy Project (CGAP)

- Cancer Therapy Evaluation Program (CTEP)

- CGAP Genetic Annotation Initiative (GAI)

- Cancer Molecular Analysis Project (CMAP)

- Affymetrix, Agilent and Illumina Microarrays

- Genome Sequencing Center at University of California, Santa Cruz (UCSC)

- SNP Annotations from NCBI

- Protein Data from Uniprot-Swissprot

- SNP Annotations from SNP Consortium

- NCI's Cancer Gene Index Data Project

- NCBI's MapViewer data

- Ensembl Compara

- DrugBank

- UniSTS

- HUGO

- and others

While this information is, in theory, available from multiple public sites, the number of links to traverse and the extent of collation that would have to be performed is daunting. The CGAP, CMAP, and GAI web sites have distilled this information from both internal and public databases, and the caBIO data warehouses have optimized it for access with respect to the types of queries defined in the APIs.

While the caBIO data are extracted from many sources that include information from a wide variety of species, we emphasize that only genomic data pertaining to human and mouse are available from caBIO.

caBIO provides access to curated data from both internal (NCI) and external sites. Table 6.3 contains data source internal to NCI and Table 6.4 contains data sources from sites outside of NCI.

NCI Data Source	Description
CGAP	CGAP (Cancer Genome Anatomy Project) provides a collection of gene expression profiles of normal, pre-cancer, and cancer cells taken from various tissues. The CGAP interface allows users to browse these profiles by various search criteria, including histology type, tissue type, library protocol, and sample preparation methods. The goal at NCI is to exploit such expression profile information for the advancement of improved detection, diagnosis, and treatment for the cancer patient. Researchers have access to all CGAP data and biological resources for human and mouse, including ESTs, gene expression patterns, SNPs, cluster assemblies, and cytogenetic information.
	The CGAP web site provides a powerful set of interactive data-mining tools to explore these data, and the caBIO project was initially conceived as a programmatic interface to these tools and data. Accordingly, most of the data that are available from CGAP can also be accessed through the caBIO objects. Exceptions are those data sets having proprietary restrictions, such as the Mitleman Chromosome Aberration database.
	CGAP also provides access to lists of sequence-verified human and mouse cDNA IMAGE clones supplied by Invitrogen.
CMAP	CMAP (Cancer Molecular Analysis Project) is powered by caCORE. The goal of CMAP is to enable researchers to identify and evaluate molecular targets in cancer.
	The CMAP Profile Query tool finds genes with the highest or lowest expression levels (using SAGE and microarray data) for a given tissue and histology. Selecting a gene from the resulting table then leads to a Gene Info page. This page provides information about cytogenetic location, chromosome aberrations, protein similarities, curated and computed orthologs, and sequence-verified as well as full-length MGC clones, along with links to various other databases.

Table 5.3 NCI-related data sources in the caBIO database

NCI Data Source	Description
CTEP	CTEP (Cancer Therapy Evaluation Program) funds an extensive national program of basic and clinical research to evaluate new anti-cancer agents, with a particular emphasis on translational research to elucidate molecular targets and drug mechanisms. In response to this emergent need for translational research, there has been a groundswell of translational support tools defining controlled vocabularies and registered terminologies to enhance electronic data exchange in areas that have heretofore been relatively non-computational. The caCORE trials data are updated with new CTEP data on a quarterly basis, and many of the objects are designed to support translational research.
	For example, a caCORE Target object represents a molecule of special diagnostic or therapeutic interest for cancer research, and an Anomaly object is an observed deviation in the structure or expression of a Target. An Agent is a drug or other intervention that is effective in the presence of one or more specific Targets. The ClinicalTrialProtocol object organizes administrative information pertaining to that protocol.
	Data from CTEP are used to populate the Protocols, ProtocolAgents and ProtocolDiseases objects.
GAI	GAI (CGAP Genetic Annotation Initiative) is an NCI research program to explore and apply technology for identification and characterization of genetic variation in genes important in cancer. The GAI utilizes data-mining to identify "candidate" variation sites from publicly available DNA sequences, as well as laboratory methods to search for variations in cancer-related genes. All GAI candidate, validated, and confirmed genetic variants are available directly from the GAI web site, and all validated SNPs have been submitted to the NCBI dbSNP database as well that are in turn available through the SNP .
HomoloGene	HomoloGene is an NCBI resource for curated and calculated gene homologs. The caBIO data sources capture only the calculated homologs stored by HomoloGene. These calculated homologs are the result of nucleotide sequence comparisons performed between each pair of organisms represented in UniGene clusters.
	caBIO provides this information via HomologousAssociation and updates this data on a monthly basis.
PID	The Pathway Interaction Database is a highly-structured, curated collection of information about known biomolecular interactions and key cellular processes assembled into signaling pathways. It is a collaborative project between the US National Cancer Institute (NCI) and Nature Publishing Group (NPG), and is an open access online resource.

Table 5.3 NCI-related data sources in the caBIO database (Continued)

NCI Data Source	Description
LocusLink/ Entrez Gene	LocusLink contains curated sequence and descriptive information associated with a gene. Each entry includes information about the gene's nomenclature, aliases, sequence accession numbers, phenotypes, UniGene cluster IDs, OMIM IDs, gene homologies, associated diseases, map locations, and a list of related terms in the Gene Ontology Consortium's ontology. Sequence accessions include a subset of GenBank accessions for a locus, as well as the NCBI Reference Sequence. The LocusLink Identifier corresponding to a caBIO Gene is available in DatabaseCrossReference.
MapView	Entrez Genomes presents a unified graphical view of maps (genetic and physical) and sequence data for a selected organism. The Entrez Map Viewer, is a software component of Entrez Genomes which provides an organism's complete genome, integrated maps (when available) for each chromosome, and sequence data for a region of interest. Data from MapView is used to populate GenePhysicalLocation and MarkerPhysicalLocation objects.
dbSNP	In collaboration with the National Human Genome Research Institute, the NCBI has established the dbSNP database to serve as a central repository for both single base nucleotide substitutions and short deletion and insertion polymorphisms. Once discovered, these polymorphisms could be used by additional laboratories, using the sequence information around the polymorphism and the specific experimental conditions. (Note that dbSNP takes the looser 'variation' definition for SNPs, so there is no requirement or assumption about minimum allele frequency.) The data from dbSNP is updated approximately every 3-4 months. Relevant information is available through SNP, Provenance, Source, URLSourceReference, SourceReference, PhysicalLocation , Location, SNPPhysicalLocation, SNPCytogeneticLocation, GeneRelativeLocation and MarkerRelativeLocation objects.
UniSTS	Database at the National Center for Biotechnology Information providing "a unified, non-redundant view of sequence tagged sites (STS's). UniSTS integrates marker and mapping data from a variety of public resources." Used in eGenome as a source of STS's, and to collect and manage element names. Data from UniSTS is used to populate Marker and MarkerAlias objects.

Table 5.3 NCI-related data sources in the caBIO database (Continued)

NCI Data Source	Description
UniGene	Unigene provides a nonredundant partitioning of the genetic sequences contained in GenBank into gene clusters. Each cluster has a unique UniGene ID and a list of the mRNA and EST sequences that are included in that cluster. Related information stored with the cluster includes tissue types in which the gene has been expressed, mapping information, and the associated LocusLink, OMIM, and HomoloGene IDs, thus providing access to related information in those NCBI databases as well through DatabaseCrossReference.
	Because the information in UniGene is centered around genes, access to Unigene is provided via the caBIO Gene objects. Specifically, the method getClusterId associated with a Gene object can be used to fetch the gene's UniGene ID. The corresponding IDs to cross-reference these genes into the NCBI OMIM and LocusLink databanks, Enzyme Commission, Ensembl and RefSeq databases can be obtained from DatabaseCrossReference using the getDatabaseCrossReferenceCollection method. While there is no explicit caBIO object corresponding to a Unigene cluster, all of the information associated with the cluster is available directly via the caBIO Gene object's methods.
	The sequences are available via NucleicAcidSequence and the associated aliases are available in the GeneAlias. Corresponding clone library and its location information is exposed via Clone and CloneRelativeLocation.
	Appropriate 'provenance' information is available in Provenance, Source, URLSourceReference and SourceReference objects.
	Markers associated with a Gene are available through the Marker Object whereas its chromosomal and cytogenetic start-stops are available through GenePhysicalLocation and GeneCytogeneticLocation objects.
	Associated Histopathological and Pathway information are available through Histopathology and Pathway respectively.
	Unigene data is updated on a monthly basis.
Cancer Gene Data Curation Project	The Cancer Gene Data Curation Project is an attempt to create a database of associations between genes and diseases and genes and drug compounds derived from the biomedical literature. The project involves a mixture of automatic text mining, semi-automatic verification, and manual validation/scoring of results.
	Data from this project is exposed through caBIO's `Evidence`, `EvidenceCode`, `GeneDiseaseAssociation` and `GeneAgentAssociation` objects

Table 5.3 NCI-related data sources in the caBIO database (Continued)

External data sources in the caBIO database are included in the following table:

External Data source	Description
Affymetrix	Affymetrix provides the majority of Microarray data for caBIO. The data provides information on allele frequencies of the SNP in different populations, and is represented by the PopulationFrequency object. The probeset information is available through ArrayReporter, ExpressionArrayReporter and SNPArrayReporter objects, amongst others. Details regarding the arrays exposed in caBIO are available through Microarray The GeneRelativeLocation object provides the location (intron, upstream, downstream etc.) of a SNP referenced in SNP Array datasets with respect to its associated genes. The validation status for a SNP comes from NCBI. The SNP Consortium (TSC) Ltd. a non-profit foundation provides the TSC id's for SNPs in DatabaseCrossReference.
Agilent	Data from Agilent's Whole Human Genome 44K Arrays and aCGH 244K Arrays are exposed through ArrayReporter, MicroArray and ExpressionArrayReporter. Associated Interpro Protein Domains and Unigene Genes are available through ProteinDomain and Gene objects.
BioCarta	BioCarta and its Proteomic Pathway Project (P3) provide detailed graphical renderings of pathway information concerning adhesion, apoptosis, cell activation, cell signaling, cell cycle regulation, cytokines/chemokines, developmental biology, hematopoeisis, immunology, metabolism, and neuroscience. NCI's CMAP web site captures pathway information from BioCarta, and transforms the downloaded image data into Scalable Vector Graphics (SVG) representations that support interactive manipulation of the online images. The CMAP web site displays BioCarta pathways selected by the user and provides options for highlighting anomalies, which include under- or over expressed genes as well as mutations. The pathway information is available via the Pathway object in caBIO. caBIO provides a class for manipulating SVG diagrams, which is described in SVG Manipulator on page 63.
DrugBank	The Canada DrugBank is a repository containing detailed information on drugs (chemical, pharmaceutical, etc.) and drug targets (sequence, pathways, etc.). The repository contains over 4800 drug entries and 2500 drug targets and is available for download as open source.
Ensembl Compara	DNA alignments across species can reveal the extent of evolutionary conservation in genetic material. The Ensembl Compara project runs many such alignment analyses and provides the results in various accessible formats. caBIO provides results from Compara's Pecan-based alignment analyses in particular, through the MultipleAlignment and related objects.

Table 5.4 External data sources in the caBIO database

External Data source	Description
[Gene Ontology Consortium](#)	The Gene Ontology Consortium provides a controlled vocabulary for the description of molecular functions, biological processes, and cellular components of gene products. The terms provided by the consortium define the recognized attributes of gene products and facilitate uniform queries across collaborating databases. In general, each gene is associated with one or more biological processes, and each of these processes may in turn be associated with many genes. In addition, the GO ontologies define many parent/child relationships among terms. For example, a branch of the ontology tree under biological process contains the term "cell cycle control", which in turn bifurcates into the "child" terms cell cycle arrest, cell cycle checkpoint, control of mitosis, etc. caBIO does not extract ontology terms directly from the Gene Ontology Consortium but, instead, extracts those terms stored with the LocusLink entry for that gene. This information is available via GoClosure, GoGenes, GeneOntology and GeneOntologyRelationship objects.
[Illumina](#)	One of the SNP-Arrays in caBIO. Data from Illumina is used to populate MicroArray and ExpressionArrayReporter, amongst others.
[SNP Consortium](#)	The SNP Consortium Ltd. is a non-profit foundation organized for providing public genomic data. Its mission is to develop up to 300,000 SNPs distributed evenly throughout the human genome and to make the information related to these SNPs available to the public without intellectual property restrictions. The TSC Ids corresponding to a SNP from dbSNP are available through DatabaseCrossReference.
[UCSC Genome](#)	UCSC (University of California, Santa Cruz Distributed Annotation System) provides the data for the Chromosomal start and end positions of mRNA, EST and Cytoband sequences. This data is used to populate the PhysicalLocation, Location, Cytoband, CytogeneticLocation and Location objects with the locations of ESTs, MRNAs and Cytobands respectively. The ESTs and MRNAs in PhysicalLocation and Location are linked with the corresponding sequence identifiers from NucleicAcidSequence
[UniProt](#) [PIR](#)	Universal Protein Resource (UniProt) is a complete annotated protein sequence database and is a central repository of protein sequence and function created by joining the information contained in Swiss-Prot, TrEMBL, and PIR. The UniProt Knowledge base provides access to extensive curated protein information, including the amino acid sequence, protein name or description, taxonomic data and protein aliases. caBIO exposes information from the Swissprot databanks through ProteinSequence and ProteinAlias objects. Mappings to RefSeq Ids are available through DatabaseCrossReference. Provenance-related information is available through Provenance, Source, SourceReference and URLSourceReference objects. Protein Domain information from Interpro is exposed through ProteinDomain object

Table 5.4 External data sources in the caBIO database (Continued)

Note:	To augment existing information on drugs and drug targets obtained from the Cancer Gene Index project, caBIO provides access to additional drug information and drug targets through the caBIO API by cross-referencing against the PharmGKB drug database.

caGrid Identifiers

The functionality provided by caGrid's Identifier Services Framework and the integration in caBIO is to have "identifiers" for each caBIO domain object. The identifier is essentially a forever globally unique name for the caBIO data-object such that it can be unambiguously used to refer to the data from different application contexts.

This identifier, obtained from the caGrid Identifier Service Framework, is essentially a string and a forever globally unique name for the caBIO domain object. Furthermore, the identifier can be (globally) resolved to the associated caBIO domain object.

In order to abstract the identifier's object properties, the data service implementations and the resolution mechanisms, the identifier's value must be treated as a "meaningless" opaque string by the consumer applications. Any leaking of design and implementation choices for the identifier framework in the applications is undesirable from an architecture point of view as it makes the implementations brittle and susceptible to future changes. Of course resolution information will have to be embedded in identifier name, but this should only be meaningful for resolution service related components that are layered below the application.

Classes Indexed for FreeStyleLM Search

The FreestyleLM (Freestyle Lexical Mine) search component provides interfaces and APIs for conducting horizontal searches across caBIO domain objects and is described in greater detail in *FreestyleLM* on page 83.

FreeStyleLM is built on Hibernate and Apache Lucene full-text search engines, providing full text search (Google™-like search) capabilities to the caBIO API. The classes and fields that are indexed are annotated in the caBIO domain model. These attributes are usually non-numeric fields with descriptions or identifiers. For example, searching for `brca1` will return any object that includes the `brca1` gene in its attributes, such as Genes, Proteins, and Pathways.

The following classes are indexed as part of the FreeStyleLM indexing application:

Class	Attribute
Agent	Id Absorption Biotransformation ChemicalFormula Comment DrugbankAccession EVSId Indication IUPACName MechanismOfAction Name Pharmacology Source Toxicity
AgentAlias	Id Name
Anomaly	Id Description
Chromosome	Id Number
ClinicalTrialProtocol	Id CurrentStatus DocumentNumber LeadOrganizationId LeadOrganizationName NIHAdminCode ParticipationType PDQIdentifier Phase Title
Condition	Name
Cytoband	Id Name
DiseaseOntology	Id Name EVSId
EntityAccession	Accession Database

Table 5.5 Classes indexed in FreeStyleLM

Class	*Attribute*
EntityName	Name
EvidenceCode	Id EvidenceCode
ExpressionArrayReporter	Id TargetDescription
Gene	Id ClusterId FullName Symbol
GeneAlias	Id Name Type
GeneOntology	Id Name
GeneFunctionAssociation	Id Role
Histopathology	Id AgeOfOnset Comments GrossDescription MicroscopicDescription
Library	Id CloneProducer CloneVector CloneVectorType Description Keyword LabHost Name UniGeneId
MacroProcess	Name
Microarray	Id Manufacturer Name Type
MultipleAlignment	Id Name Source Type

Table 5.5 Classes indexed in FreeStyleLM (Continued)

Class	*Attribute*
OrganOntology	Id
	Name
Pathway	Id
	Description
	DisplayValue
	Name
	Curator
	Reviewer
	Source
PhysicalParticipant	activityState
	location
	postTranslationalMod
PhysicalProperties	activityState
	location
	postTranslationalMod
ProteinAlias	Id
	Name
ProtocolAssociation	Id
	CTEPName
	DiseaseCategory
	DiseaseSubCategory
Protein	Id
	Keywords
	Name
	PrimaryAccession
	SecondaryAccession
	UniProtCode
Protocol	Id
	Description
	Name
	Type
ProteinDomain	Id
	AccessionNumber
	Description
	Source

Table 5.5 Classes indexed in FreeStyleLM (Continued)

Class	*Attribute*
Tissue	Id
	CellLine
	CellType
	Description
	DevelopmentalStage
	Histology
	Name
	Organ
	Supplier
	Type
Taxon	Id
	Abbreviation
	CommonName
	EthnicityStrain
	ScientificName
Target	Id
	Name
	Type
Vocabulary	Id
	CoreTerm
	GeneralTerm

Table 5.5 Classes indexed in FreeStyleLM (Continued)

caBIO Utilities

This chapter describes additional utilities provided by the caBIO API, which are not generated by the caCORE SDK.

Topics in this chapter include:

- *FreestyleLM* on this page

- *Grid Id Resolver* on page 86

- *Genome Range Queries* on page 87

- *Array Annotation API* on page 90

- *SVG Manipulator* on page 94

FreestyleLM

All of the search methods discussed in the previous chapter allow for vertical searching in one caBIO domain object. In other words, one domain object must be selected to be searched. The FreestyleLM (Freestyle Lexical Mine) search component provides and interfaces and APIs for conducting horizontal searches across caBIO domain objects.

FreestyleLM is build upon Hibernate and Apache Lucene full text search engines, providing full text search (Google™-like search) capabilities to the caBIO API. The classes and fields that are indexed are annotated in the caBIO domain model. Usually, these attributes are non-numeric fields with descriptions or identifiers. Thus, searching for "brca1" will return any object that makes mention of the "brca1" gene in its attributes, such as Genes, Proteins, Pathways, and so on.

The IndexGenerator utility within the caBIO API generates index files for these annotations. FreestyleLM provides the ability to query these indexes. The FreestyleLM search is facilitated by the classes listed below:

- `gov.nih.nci.search.SearchQuery`

- `gov.nih.nci.search.SearchResults`

- `gov.nih.nci.search.Sort`

- `gov.nih.nci.search.RangeFilter`

The search term is set as a keyword within the SearchQuery object. One or more words can be set as a keyword. Refer to the Lucene documentation for more information on the search syntax (http://lucene.apache.org/java/docs/queryparsersyntax.html).

Freestyle LM Java Client

The `CaBioApplicationService` provides a special data access method to perform the text based search.

Method Signature	`List search(SearchQuery searchQuery)`
Search Type	FreestyleLM Search
Description	Returns a List collection containing objects conforming to the criteria defined by the searchQuery.
Example	`Search(searchQuery);`

Example One: Full Text Search (Default)

In this example, a search is performed on the term 'brca*'. The caBIO FreestyleLM returns a set of SearchResults objects. The result set includes references to objects of different types that match the search term. In this case, the search results consist of information regarding Gene, Protein, Pathway, Tissue, GeneOntology and Library objects.

```
1 CaBioApplicationService appService =
      (CaBioApplicationService)ApplicationServiceProvider.getApplicationService();
2 try {
3     SearchQuery query = new SearchQuery();
4     query.setKeyword("brca*");
5     List results = appService.search(query);
6     for(int i=0; i<results.size(); i++) {
7         SearchResult result = (SearchResult)results.get(i);
8         System.out.println("Class: "+ result.getClassName() +"\t"+ result.getId()
9     }
10    System.out.println("Results: "+ results.size());
11 } catch(Exception e) {
12     e.printStackTrace();
13 }
```

Lines	*Description*
1	Create a CaBioApplicationService instance.
3-4	Instantiate a SearchQuery object and set the keyword to 'brca*'
5	Call the CaBioApplicationService search method
6-9	Iterate through the results

Example Two: Hibernate Search

A search is performed on the term 'brca*' and the query type is specified as 'HIBERNATE_SEARCH'. (The default query type is FULL_TEXT_SEARCH). The caBIO FreestyleLM result set will include objects of different types that match the search term. In this case, the search results consist of Gene, Protein, Pathway, Tissue, GeneOntology and Library objects.

```
1 CaBioApplicationService appService =
     (CaBioApplicationService)ApplicationServiceProvider.getApplicationService();
2 try {
3     SearchQuery query = new SearchQuery();
4     query.setKeyword("brca*");
5     query.setQueryType("HIBERNATE_SEARCH");
6     List results = appService.search(query);
7     for(int i=0; i<results.size(); i++){
8         Object result = (Object)results.get(i);
9         System.out.println("Class: "+ result.getClass().getName());
10    }
11    System.out.println("Results: "+ results.size());
12 } catch(Exception e){
13   e.printStackTrace();
14 }
```

Example Three: Sort Results

The result set can be sorted by specifying a sort value in the SearchQuery.

```
1 CaBioApplicationService appService =
     (CaBioApplicationService)ApplicationServiceProvider.getApplicationService();
2 try {
3     SearchQuery query = new SearchQuery();
4     query.setKeyword("brca*");
5     Sort sorter = new Sort();
6     sorter.setSortByClassName(true);
7     query.setSort(sorter);
8     List results = appService.search(query);
9     for(int i=0; i<results.size(); i++){
10        SearchResult result = (SearchResult)results.get(i);
11        System.out.println("Class: "+ result.getClassName() +"\t"+ result.getId(
12    }
13    System.out.println("Results: "+ results.size());
14 } catch(Exception e) {
15   e.printStackTrace();
16 }
```

Freestyle LM Web Interface

The caBIO API provides a simple web interface to perform text based queries. This user interface is built on the FreestyleLM Search API (*Figure 6.1*). Use the following URL to access the servlet: http://cabioapi.nci.nih.gov/cabio43/search

Figure 6.1 Freestyle Lexical Mine interface

Grid Id Resolver

Each caBIO object can be uniquely identified (even across classes) using the Grid Id (bigid attribute). See *caGrid Identifiers* on page 78 for more details about Grid Identifiers. The API discussed here allows for simple retrieval of objects using these identifiers.

Grid Id Resolver Java API

The `CaBioApplicationService` provides special data access methods to perform Grid Id searches. The same methods are available in the web services interface.

Method Signature	`boolean exist(String bigId)`
Search Type	Grid Id Resolver
Description	Check if the object specified by the given Grid Id exists in caBIO.
Example	`boolean exists =` `   exist("hdl://2500.1.PMEUQUCCL5/WX2PXQD7PB")`

Method Signature	`boolean getDataObject(String bigId)`
Search Type	Grid Id Resolver
Description	Retrieve the domain object specified by the given Grid Id.
Example	`Object obj =` `   getDataObject("hdl://2500.1.PMEUQUCCL5/WX2PXQD7PB")`

Genome Range Queries

The Range Query API provides an alternative way to search for genomic features on a chromosome, based on the physical locations. The API is capable of performing arbitrary range searches on any given chromosome (absolute search), or range searches around a feature of interest (relative search).

The caBIO model contains a subclass of `PhysicalLocation` for every object type which has `PhysicalLocation` data. For example, there is a `GenePhysicalLocation` class for Gene location data. The Range Query API takes advantage of these classes by allowing the user to specify which subclass of `PhysicalLocation` to search for. If `SNPPhysicalLocation` is specified as the target class, then only `PhysicalLocations` pertaining to SNPs will be returned.

Range Query Java API

The `CaBioApplicationService` provides a special data access methods to perform range queries.

Method Signature	`List search(RangeQuery rangeQuery)`
Search Type	Genome Range Search
Description	Returns a List collection containing objects conforming to the criteria defined by the searchQuery.
Example	`search(rangeQuery);`

Method Signature	`List search(Class targetClass, RangeQuery    rangeQuery)`
Search Type	Genome Range Search
Description	Returns a List collection containing objects of the type specified by the targetClass, conforming to the criteria defined by the given RangeQuery.
Example	`search(SNP.class,rangeQuery);`

Example One: Absolute Search

In this example, a search for SNPs on an arbitrary range of the human chromosome 1 is performed. The result set will only include SNP in the given range. By default, the reference assembly is searched. An alternative assembly may be specified with the `rangeQuery.setAssembly()` method.

```
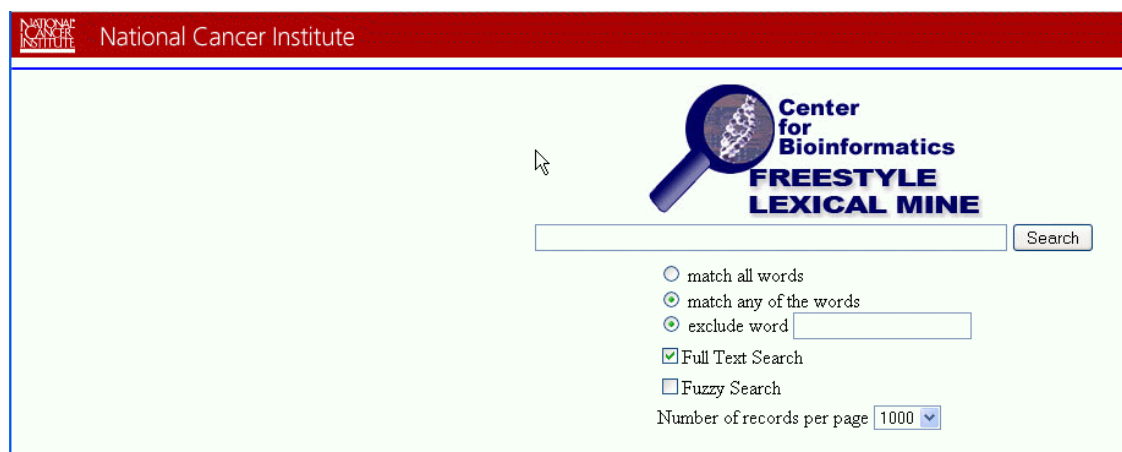1 CaBioApplicationService appService =
     (CaBioApplicationService)ApplicationServiceProvider.getApplicationService();

2  try {
3     Taxon taxon = new Taxon();
4     taxon.setAbbreviation("Hs");
5     Chromosome chromosome = new Chromosome();
6     chromosome.setNumber("1");
7     chromosome.setTaxon(taxon);
8     chromosome = (Chromosome)appService.search(
          Chromosome.class, chromosome).iterator().next();

9     AbsoluteRangeQuery query = new AbsoluteRangeQuery();
10    query.setChromosome(chromosome);
11    query.setStart(new Long(109433764));
12    query.setEnd(new Long(109434982));

13    List results = appService.search(SNPPhysicalLocation.class,query);
14    for(int i=0; i<results.size(); i++) {
15      SNPPhysicalLocation l = (SNPPhysicalLocation)results.get(i);
16      System.out.println("SNP: "+ l.getSNP().getDBSNPID());
17    }

18 } catch(Exception e) {
19    e.printStackTrace();
20 }
```

Lines	*Description*
1	Create a CaBioApplicationService instance.
3-8	Retrieve the chromosome of interest

Lines	Description
9-12	Create and configure the AbsoluteRangeQuery
13	Execute the RangeQuery with SNPPhysicalLocation as the target
14-17	Iterate through the results

Example Two: Relative Search

In this example, a search for SNPs on an arbitrary range of the human chromosome 1 is performed. The result set will only include SNP in the given range. By default, the reference assembly is searched. An alternative assembly may be specified with the `rangeQuery.setAssembly()` method.

```
1 CaBioApplicationService appService =
     (CaBioApplicationService)ApplicationServiceProvider.getApplicationService();

2  try {
3    SNP snp = new SNP();
4    snp.setDBSNPID("rs10873500");
5    snp = (SNP)appService.search(SNP.class, snp).get(0);

6    FeatureRangeQuery query = new FeatureRangeQuery();
7    query.setFeature(snp);
8    query.setUpstreamDistance(new Long(1000));
9    query.setDownstreamDistance(new Long(1000));
10   List<ArrayReporterPhysicalLocation> results = appService.search(
11       ArrayReporterPhysicalLocation.class, query);

12   for(ArrayReporterPhysicalLocation l : results) {
13     System.out.println("Reporter: "+ l.getArrayReporter().getName());
14   }
15  } catch(Exception e) {
16    e.printStackTrace();
17  }
```

Lines	Description
1	Create a CaBioApplicationService instance.
3-5	Retrieve the SNP of interest
6-9	Create and configure the FeatureRangeQuery
10-11	Execute the RangeQuery with ArrayReporterPhysicalLocation as the target
14-17	Iterate through the results

Array Annotation API

This client-side convenience API is intended for users of the Microarray annotations in caBIO. The API simplifies access to bulk microarray annotations such as reporters, genes and SNPs. The `ArrayAnnotationService` class provides the following Java methods:

Method Signature	`Map<String, Collection<String>> getGenesForReporters(String arrayPlatform, Collection<String> reporterIds)`
Description	Given an array and list of reporter names, return a mapping of those reporter names to HUGO gene symbols.
Example	`getGenesForReporters("HG-U133_Plus_2", reporterIds)`

Method Signature	`Map<String, Collection<String>> getReportersForGenes(String arrayPlatform, Collection<String> geneSymbols)`
Description	Given an array and a list of HUGO gene symbols, return a mapping of those gene symbols to reporters on the specified array.
Example	`getReportersForGenes("HG-U133_Plus_2", geneSymbols)`

Method Signature	`Collection<ArrayReporter> getReportersForPlatform(String arrayPlatform)`
Description	Returns all ArrayReporters on a given array platform.
Example	`getReportersForPlatform("HG-U133_Plus_2")`

Method Signature	`Collection<ExpressionArrayReporter> getExpressionReporterAnnotations(String arrayPlatform, Collection<String> reporterIds)`
Description	Given an array and list of reporter names, return the ExpressionArrayReporter objects corresponding to the reporter names, populated with the following annotations: ● `gene` ● `physicalLocationCollection.chromosome` ● `cytogeneticLocationCollection` ● `cytogeneticLocationCollection.cytoband`
Example	`getExpressionReporterAnnotations("HG-U133_Plus_2", reporterIds)`

Method Signature	`Collection<ExonArrayReporter>`
	`getExonReporterAnnotations(String arrayPlatform,`
	`Collection<String> reporterIds)`

Description	Given an array and list of reporter names, return the ExonArrayReporter objects corresponding to the reporter names, populated with the following annotations: ● `geneCollection` ● `physicalLocationCollection` ● `physicalLocationCollection.chromosome` ● `cytogeneticLocationCollection` ● `cytogeneticLocationCollection.cytoband`
Example	`getExonReporterAnnotations("HuEx-1_0-st-v2 ",` `reporterIds);`

Method Signature	`Collection<SNPArrayReporter>`
	`getSNPReporterAnnotations(String arrayPlatform,`
	`Collection<String> reporterIds)`

Description	Given an array and list of reporter names, return the SNPArrayReporter objects corresponding to the reporter names, populated with the following annotations: SNP ● `physicalLocationCollection` ● `physicalLocationCollection.chromosome` ● `cytogeneticLocationCollection` ● `cytogeneticLocationCollection.cytoband`
Example	`getSNPReporterAnnotations("Mapping250K_Sty",` `reporterIds)`

Method Signature	`Collection<Gene>` `getGeneAnnotations(Collection<String> geneSymbols)`
Description	Returns a collection of the requested genes, populated with the following annotations: ● `cytogeneticLocationCollection` ● `cytogeneticLocationCollection.startCytoband` ● `databaseCrossReferenceCollection` ● `chromosome`
Example	`getGeneAnnotations(geneSymbols)`

Method Signature	`List<CytobandPhysicalLocation>` `getCytobandPositions(String chromosomeNumber)`
Description	Returns all the cytobands on the specified chromosome (found on the reference assembly).
Example	`getCytobandPositions("1")`

Method Signature	`List<CytobandPhysicalLocation> getCytobandPositions(String chromosomeNumber, String assembly)`
Description	Returns all the cytobands on the specified chromosome of the given assembly.
Example	`getCytobandPositions("1", "Celera")`

Method Signature	`Microarray getMicroarray(String platformName)`
Description	Returns the Microarray with the given name.
Example	`getMicroarray("HG-U133_Plus_2")`

Method Signature	`getGenesForSymbol(String hugoSymbol)`
Description	Returns the genes with the given HUGO symbol.
Example	`getGenesForSymbol("IL5")`

Method Signature	`Collection<GeneAlias> getAliasesForGene(String symbol)`
Description	Returns the aliases associated with the given gene sumbol.
Example	`getAliasesForGene("IL5")`

Method Signature	`HashMap<String, Collection<String>> getReportersForSnps(String arrayPlatform, List<String> dbSnpIds)`
Description	Returns all reporter names for the specified SNP on the given array.
Example	`getReportersForSnps("Mapping250K_Sty", dbSnpIds)`

Method Signature	`Collection<SNP> getSNPAnnotations(List<String> dbSnpIds)`
Description	Returns the specified SNP preloaded with annotations. The associations which are preloaded: • `physicalLocationCollection` • `physicalLocationCollection.chromosome` • `cytogeneticLocationCollection` • `cytogeneticLocationCollection.startCytoband`
Example	`getSNPAnnotations(dbSnpIds)`

Method Signature	`Collection<Gene> getGenesForDbSnpId(String dbSnpId)`
Description	Returns the genes associated with the given dBSNP id by array annotations (relative locations).
Example	`getGenesForDbSnpId("rs200577")`

Method Signature	`Collection<SNP> getSnpsNearGene(String symbol, Long kbUpstream, Long kbDownstream, String assembly)`
Description	Returns all the SNPs within the specified range of the given gene on the specified assembly.
Example	`getSnpsNearGene("A1CF", 0L, 0L, "Celera")`

Method Signature	`Collection<SNP> getSnpsNearGene(String symbol, Long kbUpstream, Long kbDownstream)`
Description	Returns all the SNPs within the specified range of the given gene, on the reference assembly.
Example	`getSnpsNearGene("A1CF", 1000L, 1000L)`

Example One: Retrieve Gene Symbols for Reporters on the HGU-133 Array

In this example, a search for SNPs on an arbitrary range of the human chromosome 1 is performed. The result set will only include SNP in the given range. By default, the reference assembly is searched. An alternative assembly may be specified with the `rangeQuery.setAssembly()` method.

```
1 CaBioApplicationService appService =

  (CaBioApplicationService)ApplicationServiceProvider.getApplicationServ
  ice();
2 ArrayAnnotationService aaService = new
  ArrayAnnotationServiceImpl(appService);

3 List<String> list = new ArrayList<String>();
4 list.add("1554206_at");
5 list.add("1555490_s_at");
6 list.add("1556516_at");

7 Map<String, Collection<String>> results =
8     aaService.getGenesForReporters("HG-U133_Plus_2", list);

9  for(String reporter : results.keySet()) {
10     System.out.println(reporter+" :
  "+results.get(reporter).iterator().next());
11  }
```

Lines	Description
1	Create a CaBioApplicationService instance.
2	Create the ArrayAnnotationService instance.
3-6	Create a list of reporter names.
7-8	Execute the convenience method
9-11	Iterate through the results

Example Two: Retrieve SNPs with Pre-loaded Annotations

In this example, a search for SNPs on an arbitrary range of the human chromosome 1 is performed. The result set will only include SNP in the given range. By default, the reference assembly is searched. An alternative assembly may be specified with the caBIO 4.3 Technical Guide method.

```
1 CaBioApplicationService appService =
     (CaBioApplicationService)ApplicationServiceProvider.getApplicationService();

2   ArrayAnnotationService aaService = new ArrayAnnotationServiceImpl(appService);

3   List<String> list = new ArrayList<String>();
4   list.add("rs16893403");
5   list.add("rs1598492");
6   list.add("rs6768736");

7   Collection<SNP> results = aaService.getSNPAnnotations(list);

8   for(SNP snp : results) {
9       PhysicalLocation pl = snp.getPhysicalLocationCollection().iterator().next();
10      System.out.println(snp.getDBSNPID()+" on chromosome
"+pl.getChromosome().getNumber());
11  }
```

Lines	*Description*
1	Create a CaBioApplicationService instance.
2	Create the ArrayAnnotationService instance.
3-6	Create a list of SNP ids
7	Execute the convenience method to retrieve the SNP annotations
8	Iterate through the results
9-10	Follow preloaded associations. None of these method calls produce additional queries.

SVG Manipulator

In the `gov.nih.nci.common.util` package, the caBIO API includes a class (`SVGManipulator`) that provides useful services to manipulate Scalable Vector Graphics (SVG) diagrams retrieved from the caBIO Pathway domain object.

Manipulating Pathway Diagrams

BioCarta and its Proteomic Pathway Project (P3) provide detailed graphical renderings of pathway information. NCI's CMAP web site captures pathway information from BioCarta, and transforms the downloaded image data into Scalable Vector Graphics (SVG) representations that support interactive manipulation of the online images. The `SVGManipulator` utility class provides the capability to do the following:

- Change the display colors for each gene contained in an SVG diagram.

- Modify the URL linking a gene in the SVG diagram to external gene information. The default gene URL links to the CMAP website.

- Disable all genes or nodes within an SVG diagram.

- Retrieve a gene's color.

- Reset a gene or node to its original state.

- Retrieve/set SVG diagram attributes via getter/setter methods

SVG Diagram Manipulation Utility Example

The following test client demonstrates how to use the `SVGManipulation` class to change the appearance of a pathway diagram associated with a given pathway.

```
1    CaBioApplicationService appService =
         (CaBioApplicationService)ApplicationServiceProvider.getApplicationService();

2    Pathway pw = new Pathway();
3    pw.setName("h_freePathway");
4    List results = appService.search(Pathway.class, pw);
5    Pathway returnedPw = (Pathway) results.get(0);
6    SVGManipulator svgM = new SVGManipulator(returnedPw);

7    Gene[] genes = new Gene[2];

8    Gene gene1 = new Gene();
9    gene1.setClusterId(new Long(443914)); // SOD1
10   List resultList1 = appService.search(Gene.class, gene1);
11   genes[0] = (Gene) resultList1.get(0);

12   Gene gene2 = new Gene();
13   gene2.setClusterId(new Long(241570)); // TNF
14   List resultList2 = appService.search(Gene.class, gene2);
15   genes[1] = (Gene) resultList2.get(0);

16   String[] colors = new String[2];
17   colors[0] = "255,0,0"; // SOD1 = red
18   colors[1] = "0,0,255"; // TNF = blue
19   svgM.setSvgColors(genes, colors);

20   svgM.saveXMLDoc("out.svg", svgM.getSvgDiagram());
```

Lines	Description
2-6	Retrieve "h_freePathway" and create a new SVManipulator instance to operate on it.
7-15	Retrieve genes of interest in the pathway
16-18	Create a parallel list of colors for the genes
19	Set the colors for the genes
20	Save the modified SVG image

REFERENCES

Technical Manuals/Articles

1. National Cancer Institute. caCORE SDK 4.1 Developer's Guide http://ncicb.nci.nih.gov/NCICB/infrastructure/cacoresdk#Documentation

2. Java Bean Specification: http://java.sun.com/products/javabeans/docs/spec.html

3. Foundations of Object-Relational Mapping: http://www.chimu.com/publications/objectRelational/

4. Object-Relational Mapping articles and products: http://www.service-architecture.com/object-relational-mapping/

5. Hibernate Reference Documentation:http://www.hibernate.org/5.html

6. Basic O/R Mapping: http://www.hibernate.org/hib_docs/v3/reference/en/html/mapping.html

7. Java Programming: http://java.sun.com/learning/new2java/index.html

8. Jalopy User Manual: http://jalopy.sourceforge.net/existing/manual.html

9. Javadoc tool: http://java.sun.com/j2se/javadoc/

10. JUnit: http://junit.sourceforge.net/

11. Extensible Markup Language: http://www.w3.org/TR/REC-xml/

12. XML Metadata Interchange: http://www.omg.org/technology/documents/formal/xmi.htm

13. Ehcache: http://ehcache.sourceforge.net/documentation/

Scientific Publications

1. Ansher SS and Scharf R (2001). The Cancer Therapy Evaluation Program (CTEP) at the National Cancer Institute: industry collaborations in new agent development. Ann N Y Acad Sci 949:333-40.

2. Boon K, Osorio EC, Greenhut SF, Schaefer CF, Shoemaker J, Polyak K, Morin PJ, Buetow KH, Strausberg RL, De Souza SJ, and Riggins GJ (2002). An anatomy of normal and malignant gene expression. Proc Natl Acad Sci U S A 2002 Jul 15.

3. Buetow KH, Klausner RD, Fine H, Kaplan R, Singer DS, and Strausberg RL (2002). Cancer Molecular Analysis Project: Weaving a rich cancer research tapestry. Cancer Cell 1(4):315-8.

4. Boguski & Schuler (1995). ESTablishing a human transcript map. Nature Genetics 10: 369-71.

5. Clifford R, Edmonson M, Hu Y, Nguyen C, Scherpbier T, and Buetow KH (2000). Expression-based genetic/physical maps of single-nucleotide polymorphisms identified by the Cancer Genome Anatomy Project. Genome Res 10(8):1259-65.

6. Covitz P.A., Hartel F., Schaefer C., De Coronado S., Sahni H., Gustafson S., Buetow K. H. (2003). caCORE: A common infrastructure for cancer informatics. Bioinformatics. 19: 2404-2412.

7. Dowell RD, Jokerst RM, Day A, Eddy SR, Stein L. The Distributed Annotation System. BMC Bioinformatics 2(1):7.

8. The Gene Ontology Consortium (2000). Gene ontology: tool for the unification of biology. Nature Genetics 25:25-9.

9. The Gene Ontology Consortium. (2001). Creating the gene ontology resource: design and implementation. Genome Res 11:1425-33.

10. Hartel FW and de Coronado S (2002). Information standards within NCI. In: Cancer Informatics: Essential Technologies for Clinical Trials. Silva JS, Ball MJ, Chute CG, Douglas JV, Langlotz C, Niland J and Scherlis W, eds. Springer-Verlag.

11. Karolchik D, Baertsch R, Diekhans M, Furey TS, Hinrichs A, Lu YT, Roskin KM, Schwartz M, Sugnet CW, Thomas DJ, Weber RJ, Haussler D and Kent WJ(2003) The UCSC Genome Browser Database. Nucl. Acids Res 31(1), 51-54.

12. Paten B., Herrero J., Beal K., Fitzgerald S., and Birney Enredo E. and Pecan: Genome-wide mammalian consistency-based multiple alignment with paralogs. Genome Research 2008 18: 1814-1828.

13. Pruitt KD, Katz KS, Sicotte H, and Maglott DR (2000). Introducing RefSeq and LocusLink: curated human genome resources at the NCBI. Trends Genet 16(1):44-7.

14. Pruitt KD, and Maglott DR (2001). RefSeq and LocusLink: NCBI gene-centered resources. Nucleic Acids Res 29(1):137-40.

15. Schuler et al. (1996). A gene map of the human genome. Science 274: 540-6.

16. Schuler (1997). Pieces of the puzzle: expressed sequence tags and the catalog of human genes. J Mol Med 75(10):694-8.

17. Strausberg RL (1999). The Cancer Genome Anatomy Project: building a new information and technology platform for cancer research. In: Molecular Pathology of Early Cancer (Srivastava S, Henson DE, Gazdar A, eds. IOS Press, 365-70.

18. Strausberg RL (2001). The Cancer Genome Anatomy Project: new resources for reading the molecular signatures of cancer. J Pathol 195:31-40.

19. Sherry ST, Ward MH, Kholodov M, Baker J, Phan L, Smigielski EM, Sirotkin K (2001). dbSNP: the NCBI database of genetic variation. Nucleic Acids Res. 1;29(1):308-11.

20. Lennon G, Auffray C, Polymeropoulos M, Soares MB(1996). The I.M.A.G.E. Consortium: an integrated molecular analysis of genomes and their expression. Genomics. 33(1):151-2.

21. Wheeler DL, Barrett T, Benson DA, Bryant SH, Canese K, Chetvernin V, Church DM, DiCuccio M, Edgar R, Federhen S, Geer LY, Kapustin Y, Khovayko O, Landsman D, Lipman DJ, Madden TL, Maglott DR, Ostell J, Miller V, Pruitt KD, Schuler GD, Sequeira E, Sherry ST, Sirotkin K, Souvorov A, Starchenko G, Tatusov RL, Tatusova TA, Wagner L, Yaschenko E.(2007) Epub(2006) Database resources of the National Center for Biotechnology Information. Nucleic Acids Res. 35(Database issue):D5-12.

22. Wishart DS, Knox C, Guo AC, Cheng D, Shrivastava S, Tzur D, Gautam B, Hassanali. DrugBank: a knowledgebase for drugs, drug actions and drug targets. M. Nucleic Acids Res. 2008 Jan;36(Database issue):D901-6. PMID: 18048412.

23. Wishart DS, Knox C, Guo AC, Shrivastava S, Hassanali M, Stothard P, Chang Z, Woolsey J. DrugBank: a comprehensive resource for in silico drug discovery and exploration. Nucleic Acids Res. 2006 Jan 1;34(Database issue):D668-72. PMID: 16381955.

24. Pontius JU, Wagner L, Schuler GD (2003). UniGene: a unified view of the transcriptome In: The NCBI Handbook. Bethesda (MD): National Center for Biotechnology Information.

25. Schuler GD (1997). Pieces of the puzzle: expressed sequence tags and the catalog of human genes. J Mol Med 75:694-698.

26. Schuler GD, et al (1996). A gene map of the human genome. Science 274:540-546.

27. Thorisson GA, Stein LD (2003). The SNP Consortium website: past, present and future. Nucleic Acids Res. 31(1):124-7.

28. Zhang, Schwartz, Wagner, and Miller (2000). A Greedy algorithm for aligning DNA sequences. J Comp Biol 7(1-2):203-14.

caBIG® Material

1. caBIG®: http://cabig.nci.nih.gov/

2. caBIG® Compatibility Guidelines:
   http://cabig.nci.nih.gov/guidelines_documentation

caCORE Material

1. caCORE: https://wiki.nci.nih.gov/display/caCORE/caCORE+SDK

2. caBIO: https://wiki.nci.nih.gov/display/ICR/caBIO

3. caDSR: https://wiki.nci.nih.gov/display/caDSR/caDSR+Wiki+Home+Page

4. EVS: https://wiki.nci.nih.gov/display/EVS/EVS+Wiki

5. CSM: https://wiki.nci.nih.gov/display/caCORE/
   Common+Security+Module+%28CSM%29

Modeling Concepts

1. Enterprise Architect Online Manual: http://www.sparxsystems.com.au/
   EAUserGuide/index.html

2. OMG Model Driven Architecture (MDA) Guide Version 1.0.1:
   http://www.omg.org/docs/omg/03-06-01.pdf

3. Object Management Group: http://www.omg.org/

Applications Currently Using caBIO

1. BIO Browser: http://www.jonnywray.com/java/index.html

2. caPathway: http://cgap.nci.nih.gov/Pathways

3. GEWorkbench: https://cabig.nci.nih.gov/tools/geWorkbench

4. Rembrandt: https://caintegrator.nci.nih.gov/rembrandt/

INDEX

T

Tomcat 15
TrEMBL 77

U

UCSC 77
Unigene 75
UniProt 77
UniSTS 74
Utility Methods
    SVG Manipulation Utility 94
utility methods, XML Utility 45

W

war 15
Web service package 18
Web Services API
    endpoint URL 46
    operations 46
    WSDL file 46
web services API
    caBIOService endpoint operations 47
    configuration 46
    description 46
    Java client 46
    limitations 50
    use examples 49

X

XLinks, retrieving related results using 54
XML-HTTP API
    description 51
    limitations 54
    results sets 53
    service location 51
    syntax 51
    use examples 53
XML utility 45
XSLT 15